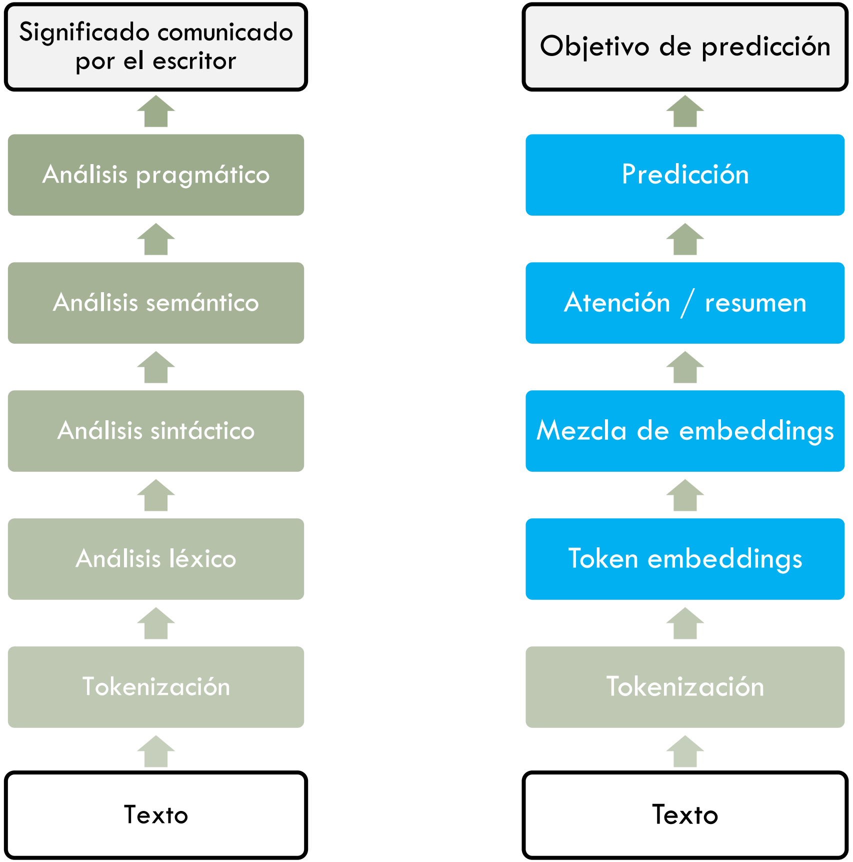
Análisis de textos:
Métodos estadísticos avanzados

Fuente diapositivas: https://github.com/albarji/curso-analisis-textos

Ainhoa Goñi Espinilla

Estratos de procesamiento en aprendizaje automático



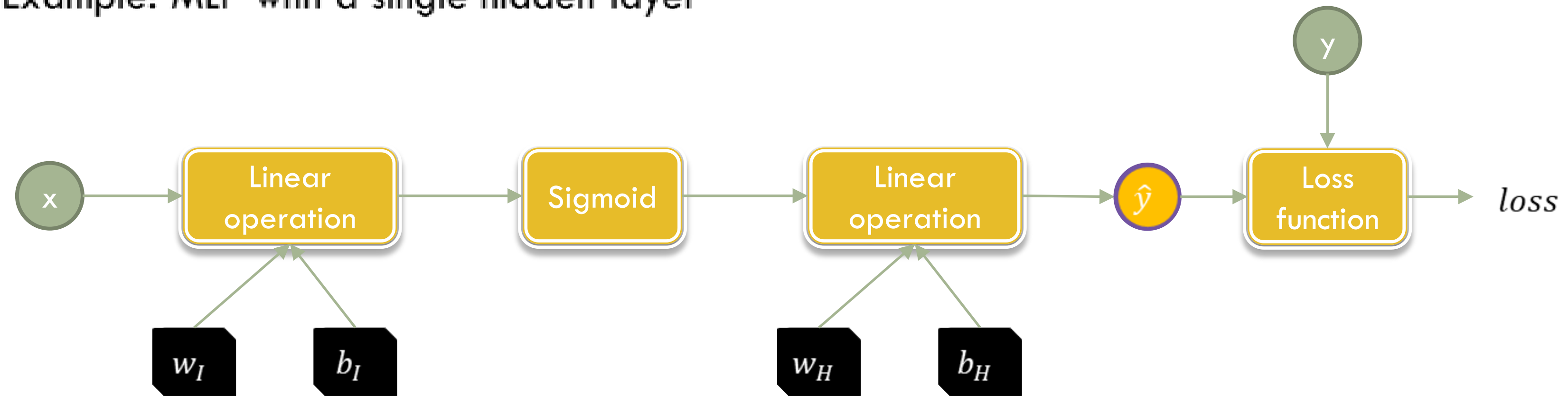
- En la práctica los niveles léxico, sintáctico y semántico están muy interrelacionados, y no es sencillo analizarlos por separado.
- Por ello en métodos actuales de análisis de texto se emplean modelos de aprendizaje automático, principalmente redes neuronales profundas, para obtener:
  - Representaciones de palabras (tokens)
  - Representaciones de frases/textos, mezclando las representaciones de palabras.
- Filosofía: no tratar de resolver el problema de lenguaje de forma general, sino solo el problema concreto que nos interesa (objetivo de predicción).

Redes neuronales para Análisis de Textos

What is really a neural network?

A neural network can be thought as an arbitrary directed acyclic graph (DAG) of operations, from an input vector x to an output vector $\hat{y}$. Each operation has some parameters to be tuned to minimize a loss function.

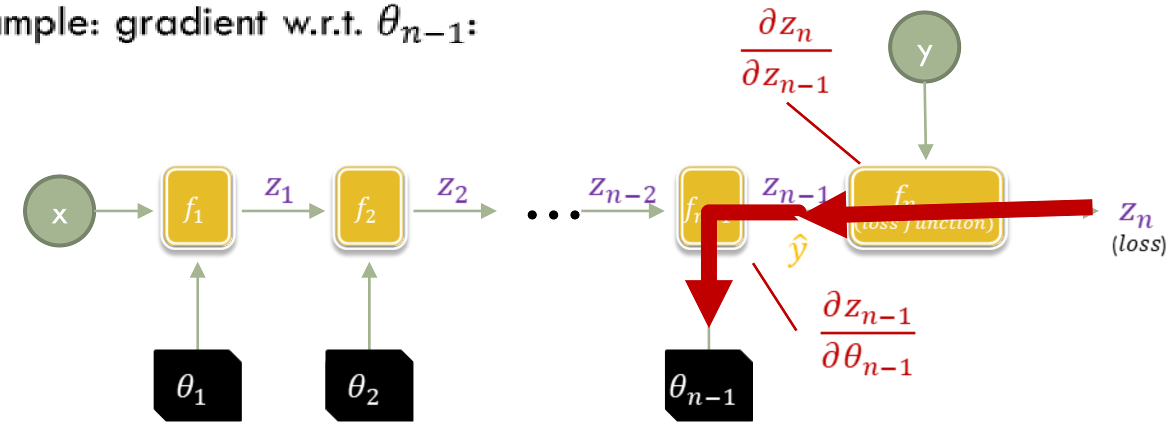
Example: MLP with a single hidden layer



What is backpropagation doing?

Visually, backpropagation travels the network backwards from $loss$ to θ_i. Each function in the path adds a partial derivative of its output with respect to its input.

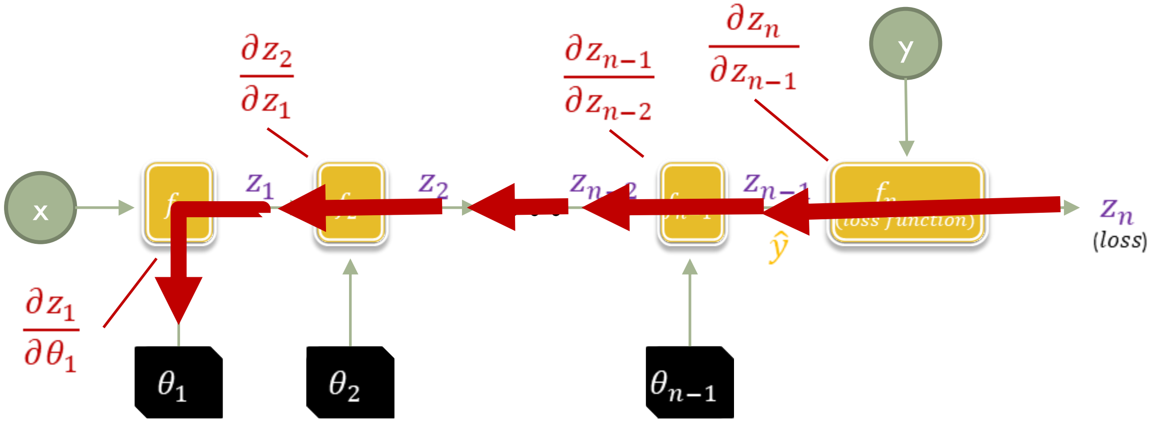
Example: gradient w.r.t. θ_{n-1}:



$$\frac{\partial F}{\partial \theta_{n-1}} = \frac{\partial z_n}{\partial z_{n-1}} \frac{\partial z_{n-1}}{\partial \theta_{n-1}}$$

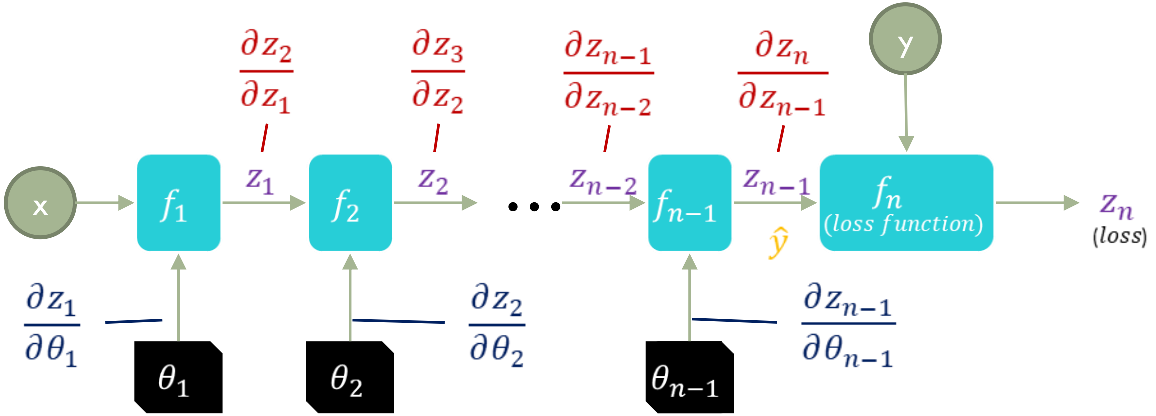
What is backpropagation doing?

Another example: gradient w.r.t. θ_1:



$$\frac{\partial F}{\partial \theta_1} = \frac{\partial z_n}{\partial z_{n-1}} \frac{\partial z_{n-1}}{\partial z_{n-2}} \cdots \frac{\partial z_2}{\partial z_1} \frac{\partial z_1}{\partial \theta_1}$$

Important observations about backpropagation



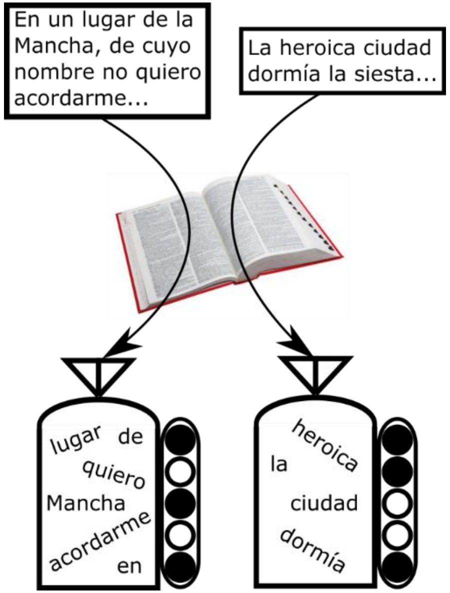
$$\frac{\partial F}{\partial \theta_{n-1}} = \frac{\partial z_n}{\partial z_{n-1}} \frac{\partial z_{n-1}}{\partial \theta_{n-1}} \qquad \frac{\partial F}{\partial \theta_{n-2}} = \frac{\partial z_n}{\partial z_{n-1}} \frac{\partial z_{n-1}}{\partial z_{n-2}} \frac{\partial z_{n-2}}{\partial \theta_{n-2}} \qquad \frac{\partial F}{\partial \theta_{n-3}} = \frac{\partial z_n}{\partial z_{n-1}} \frac{\partial z_{n-1}}{\partial z_{n-2}} \frac{\partial z_{n-2}}{\partial z_{n-3}} \frac{\partial z_{n-3}}{\partial \theta_{n-3}}$$

- Terms in the form $\frac{\partial z_i}{\partial z_{i-1}}$ can be reused throughout $\frac{\partial F}{\partial \theta_i}$ computations $\forall\, i$ → efficient computation of derivatives for all network parameters.
- We can add any function f_i to a neural network, as long as we can compute every partial derivate of each output w.r.t. each input.
- Layers closer to the input → more terms in gradient calculation

Basic feature generation: bag of words

- Build a dictionary of tokens in the training data (V tokens)
- Assign numbers $1 \ldots V$ to each token
- Encode each token as an all-zeros vector of length V, except for a 1 in the position corresponding to the token number
- Encode documents as sum of their tokens vectors (bag of words)
- Use the encoded data as inputs to the model.

Index	Token
5	.
12	the
19	cat
20	dog
…	…
288	eat
827	run
…	…



Better token representations representations

- Bag of words one-hot vectors work, but lack semantic meaning
- All words are equally apart in vector space, regardless of their meaning (semantics) or function (syntax)

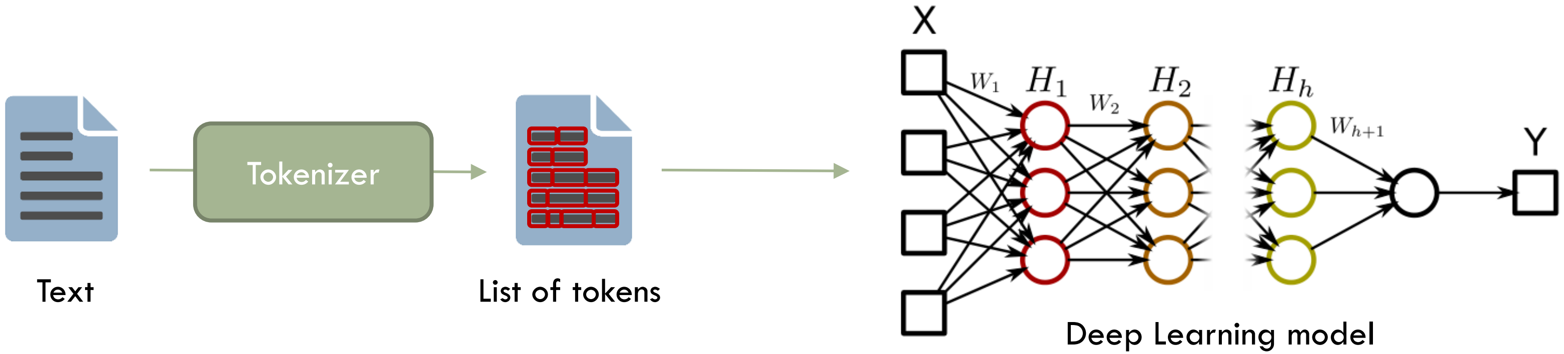
motel [0 0 0 0 0 0 0 0 0 0 1 0 0 0 0] AND
hotel [0 0 0 0 0 0 0 1 0 0 0 0 0 0 0] = 0

- Are there better encodings for words?
- Even more: can we learn them automatically?

Socher, Bengio, Manning: Deep Learning for NLP (without Magic): http://www.socher.org/index.php/DeepLearningTutorial/DeepLearningTutorial

Deep Learning text processing pipeline

Let the neural network do the feature engineering!



But there are two issues:

X Inputs to a neural network must be numbers, not strings!

X Each text in a dataset might have different number of tokens

Automatically computing features for tokens

- Encode each token as in bag-of-words (one-hot) of length N
- Multiply this vector by an embedding matrix $W \in \mathbb{R}^{Vxd}$
  - For a one-hot vector with a 1 in the i-th entry, this operation returns the i-th row of W
  - $d \equiv$ embedding dimension. $[50, 300]$ usually works well.

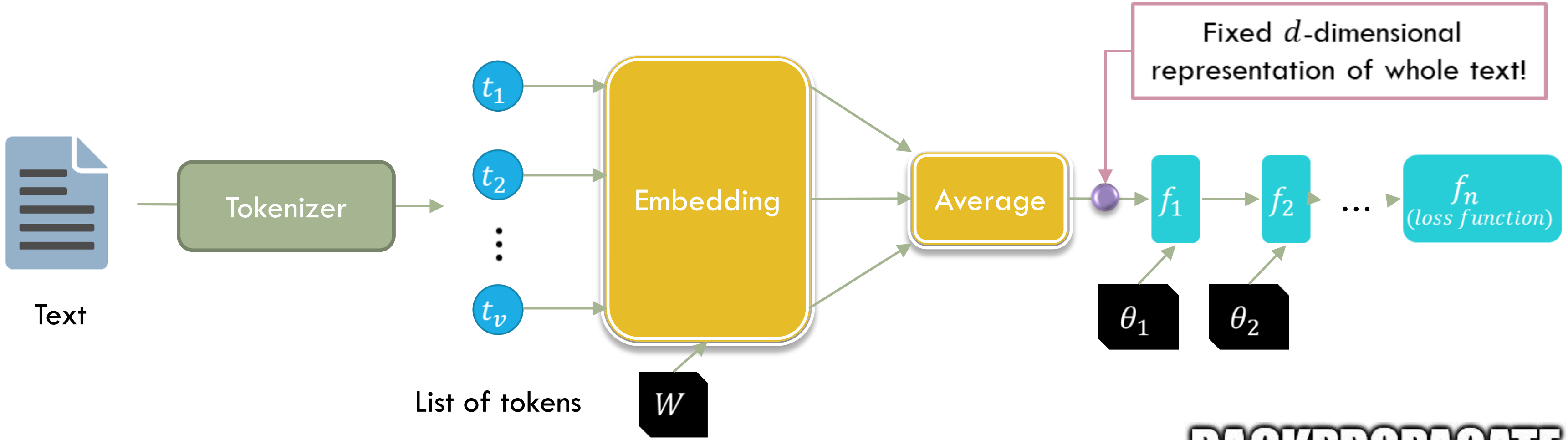
dog →

Index	Token
5	.
12	the
19	cat
20	dog
…	…
288	eat
827	run
…	…

index 20

$$\begin{bmatrix} \vdots \\ 0 \\ 1 \\ 0 \\ \vdots \end{bmatrix} \bullet W^\top = \begin{bmatrix} w_{11} & \cdots & w_{1d} \\ \vdots & \ddots & \vdots \\ w_{\hat{x}1} & \cdots & w_{\hat{x}d} \\ \vdots & \ddots & \vdots \\ w_{|V|1} & \cdots & w_{|V|d} \end{bmatrix} \rightarrow \begin{bmatrix} w_{20,1} \\ w_{20,2} \\ \vdots \\ w_{20,d} \end{bmatrix}$$

The Continuous Bag of Words model (CBoW)



- Apply the same embedding matrix W to every token in the sentence
- Average across tokens to obtain a fixed-length vector
- Use backpropagation to learn both network parameters θ_i and token representations W

CBoW performance

✓ Many practical text problems can be solved with CBoW

BUT…

X Good word representations (embeddings W) can only be learned effectively when using large training datasets.

→ solution: language models

X CBoW does not take into account word order. This might be relevant for harder problems: translation, subtle meaning, …

→ solution: mixing models

Language models

Language Models

Objective: build a model that given a sentence, returns the probability of that sentence appearing in the language.

$$p(X) = p(x_1, x_2, \dots, x_T)$$

Unsupervised learning problem: we just need large amounts of text to do it. No labels required!

- ✓ Wikipedia
- ✓ Twitter
- ✓ Webscrapping
- ✓ Any text source!

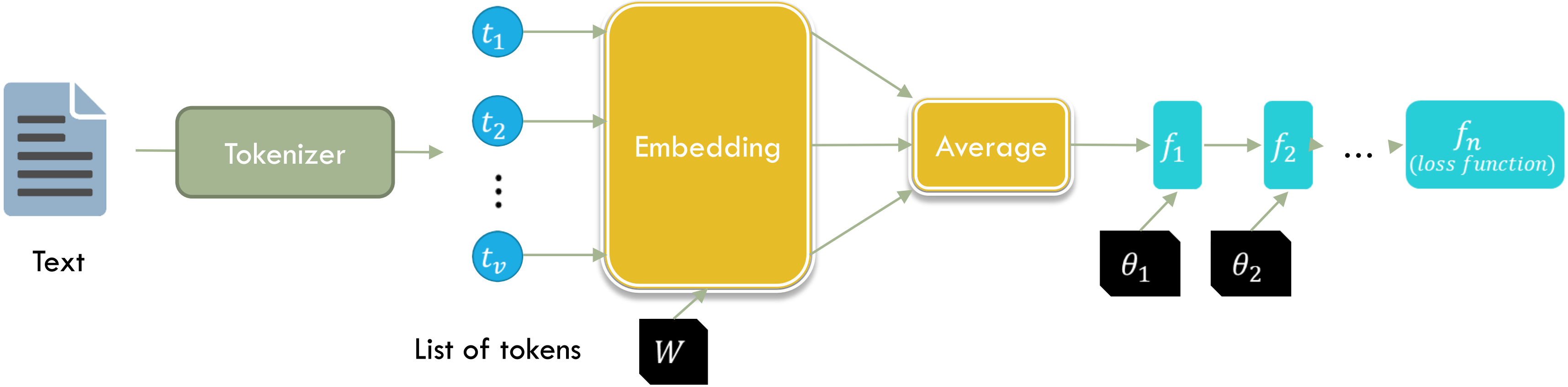
But… how do we learn $p(x_1, x_2, \dots, x_T)$?

Autoregressive Language Models

Using the definition of condicional probability

$$p(X) = p(x_1, x_2, \ldots, x_T) = p(x_1)p(x_2|x_1) \ldots p(x_T|x_1, \ldots, x_{T-1}) = \prod_{t=1}^{T} p(x_t|x_{<t})$$

The unsupervised problem is now a supervised classification problem: predict next word in the sentence → use CBoW!

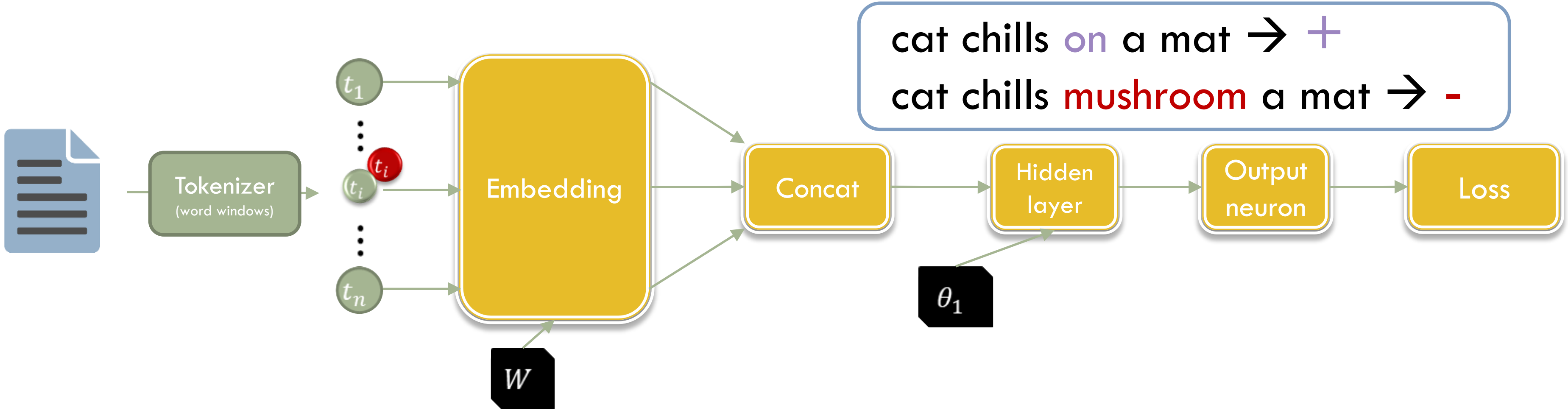


Language model example: word2vec

For each n-word window in the text, create 2 training examples:

- Positive example: concatenate all R vectors encoding the n words
- Negative example: equal to positive, but changing the central word to another random word in the training data

The language model tries to tell apart real language "sentences" from noisy ones



Socher, Bengio, Manning: Deep Learning for NLP (without Magic): http://www.socher.org/index.php/DeepLearningTutorial/DeepLearningTutorial

word2vec: learned embeddings

- Closest words to ‘france’

Word	Cosine distance
spain	0.678515
belgium	0.665923
netherlands	0.652428
italy	0.633130
switzerland	0.622323
luxembourg	0.610033
portugal	0.577154
russia	0.571507
germany	0.563291
catalonia	0.534176

- Some k-means clusters

carnivores 234	acceptance 412
carnivorous 234	argue 412
cetaceans 234	argues 412
cormorant 234	arguing 412
coyotes 234	argument 412
crocodile 234	arguments 412
crocodiles 234	belief 412
crustaceans 234	believe 412
cultivated 234	challenge 412
danios 234	claim 412

word2vec: semantic algebra

King − man + woman = queen

Obama − USA + Russia = Putin

paella − Spain + Italy = risotto

Cristiano − Madrid + Barcelona = Messi

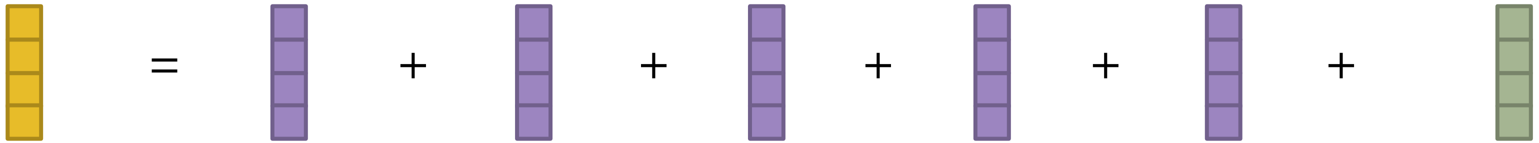
Language model example: fasttext

✓ word2vec produces better word representations

X But ignores the inner structure of words (morphology). This is relevant in languages with complex morphology: spanish, german, …

The fasttext model proposes an improvement by learning a vectors for subwords (character n-grams). The vector representing a token is the sum of its subword vectors, plus a special “subword” representing the whole token.

where = <wh + whe + her + ere + re> + <where>

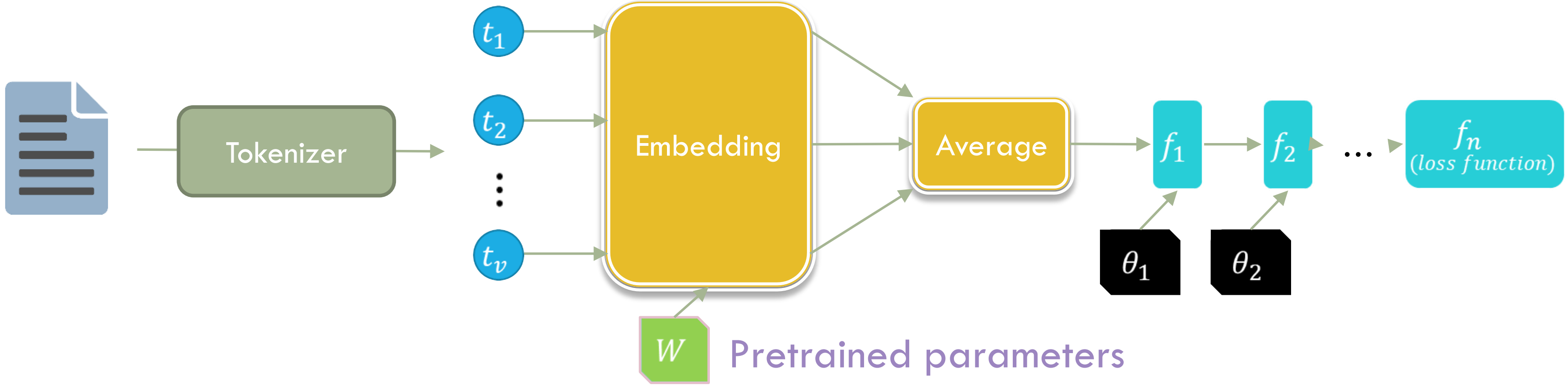


Efficient implementations allow learning a more precise language model with extremely large corpora: Common Crawl.

Bojanowski et al – Enriching Word Vectors with Subword Information

Using language models in supervised models

✓ When training data is scarce, initializing the network embeddings with the pretrained parameters from a language model can be very effective



word2vec: https://code.google.com/p/word2vec/

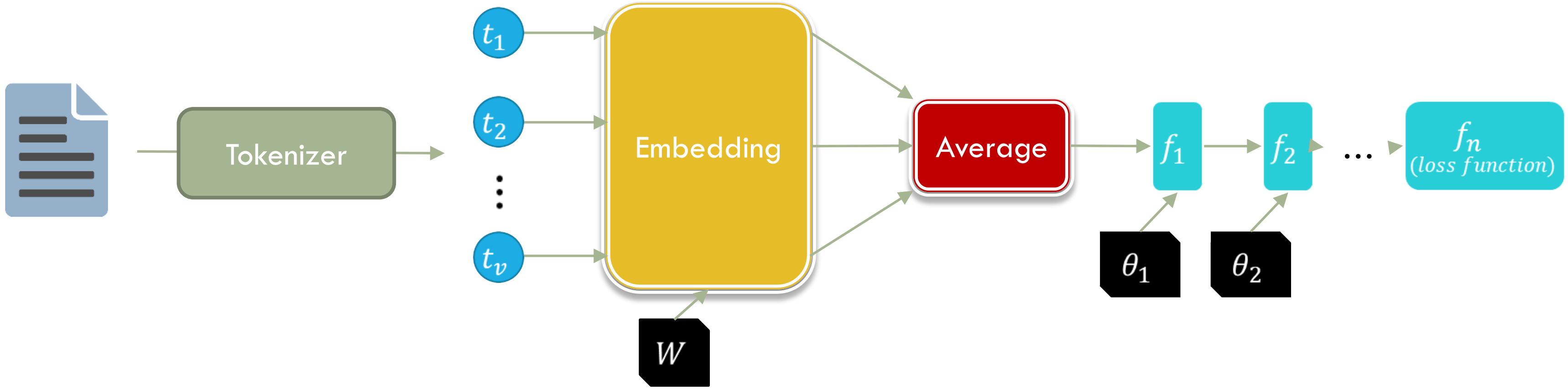
Spanish Billion Words Corpus and Embeddings: http://crscardellino.me/SBWCE/

FastText: https://fasttext.cc/

Mixing models

Better token combinations

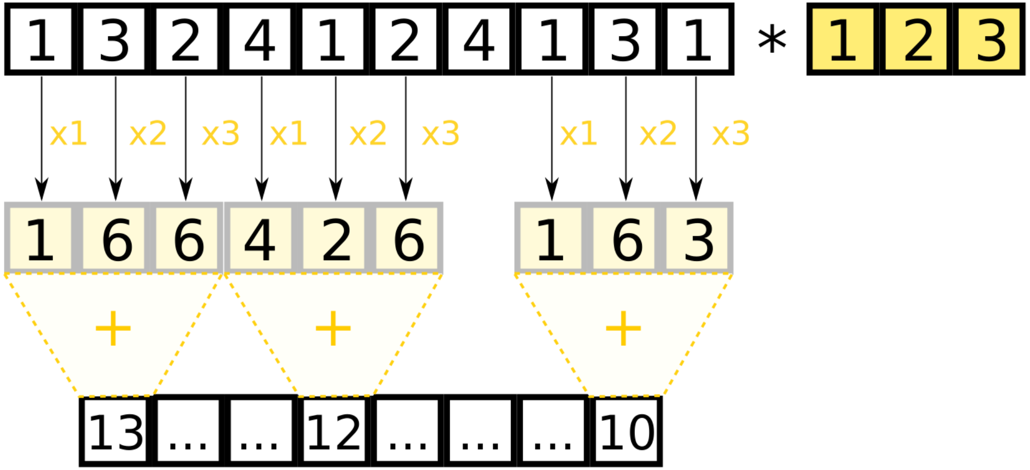
In some problems word ordering is very important. We can't just take the average of word representations!



I had my car cleaned	break the end	Me río en el baño
$\neq$	$\neq$	$\neq$
I had cleaned my car	end the break	Me baño en el río

One-dimensional convolutions

- Operation between continuous functions $$(x * z)(t) = \int_{-\infty}^{+\infty} x(a)\, z(t-a)\, \mathrm{d}a$$
- For discrete signals $$(x * z)_i = \sum_{j=-\infty}^{+\infty} x_j\, z_{i-j}$$
- Applies a small layer of weights over each portion of the input

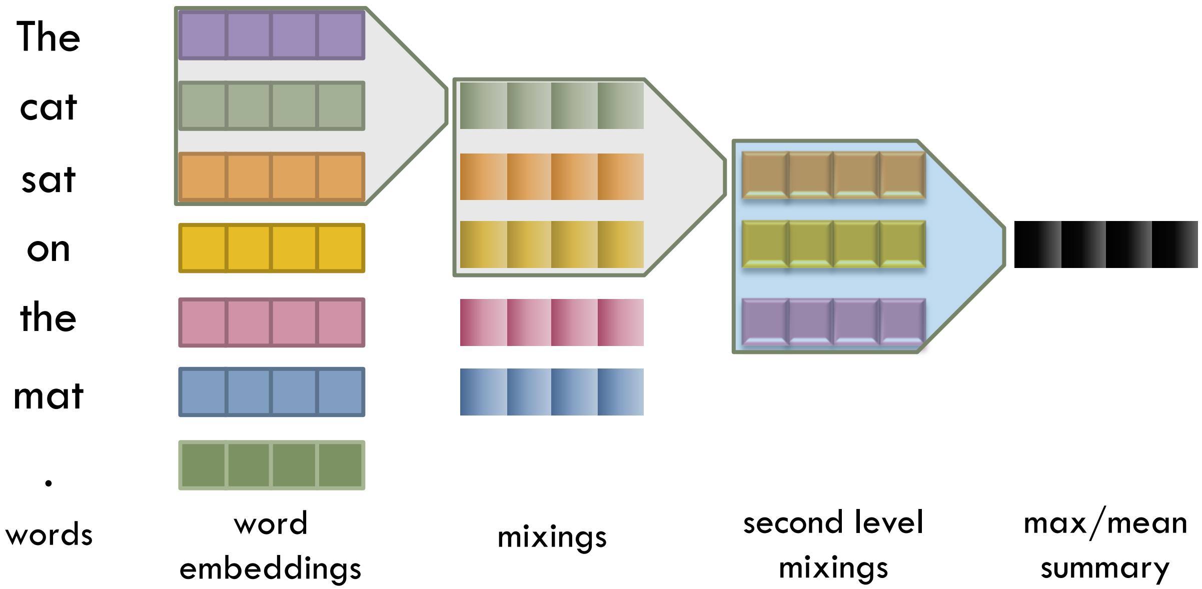


Convolutions over embeddings

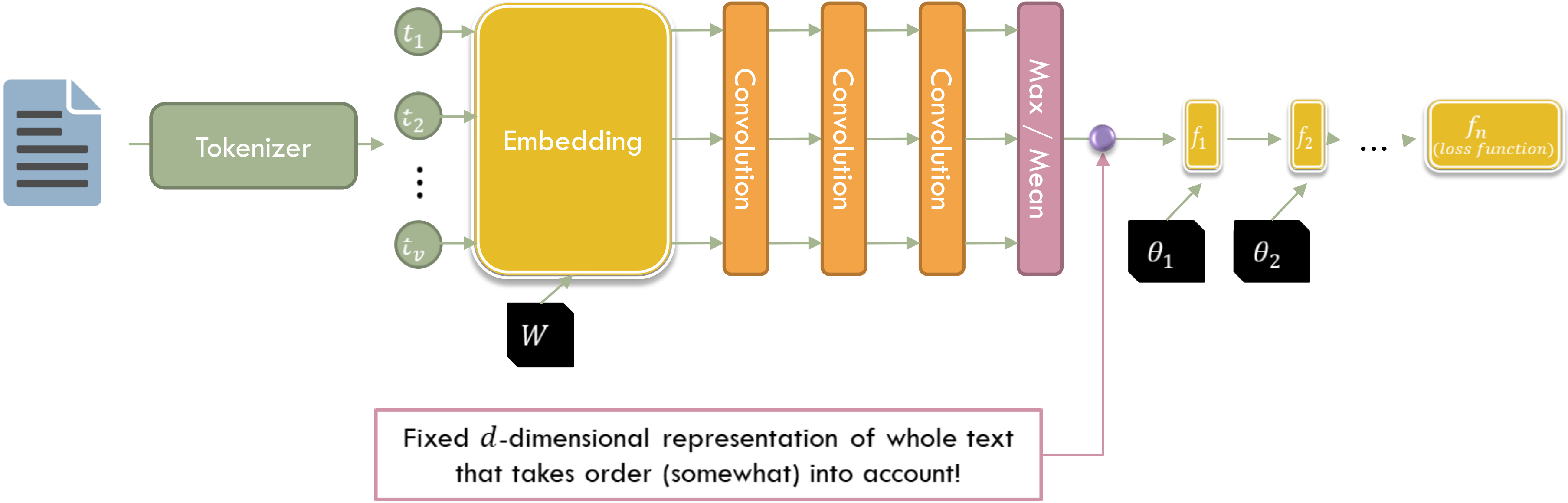
Convolutions are helpful to mix the representation of a word with the representation of neighboring words.

By stacking convolutions a more broad mixing can be obtained.

After a number of convolutions, the mean or max values are taken to obtain a single vector that sums up the whole input document.

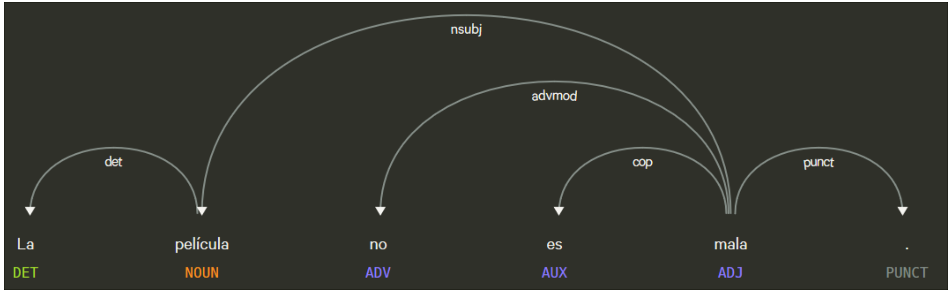


Stacked convolutions: network diagram

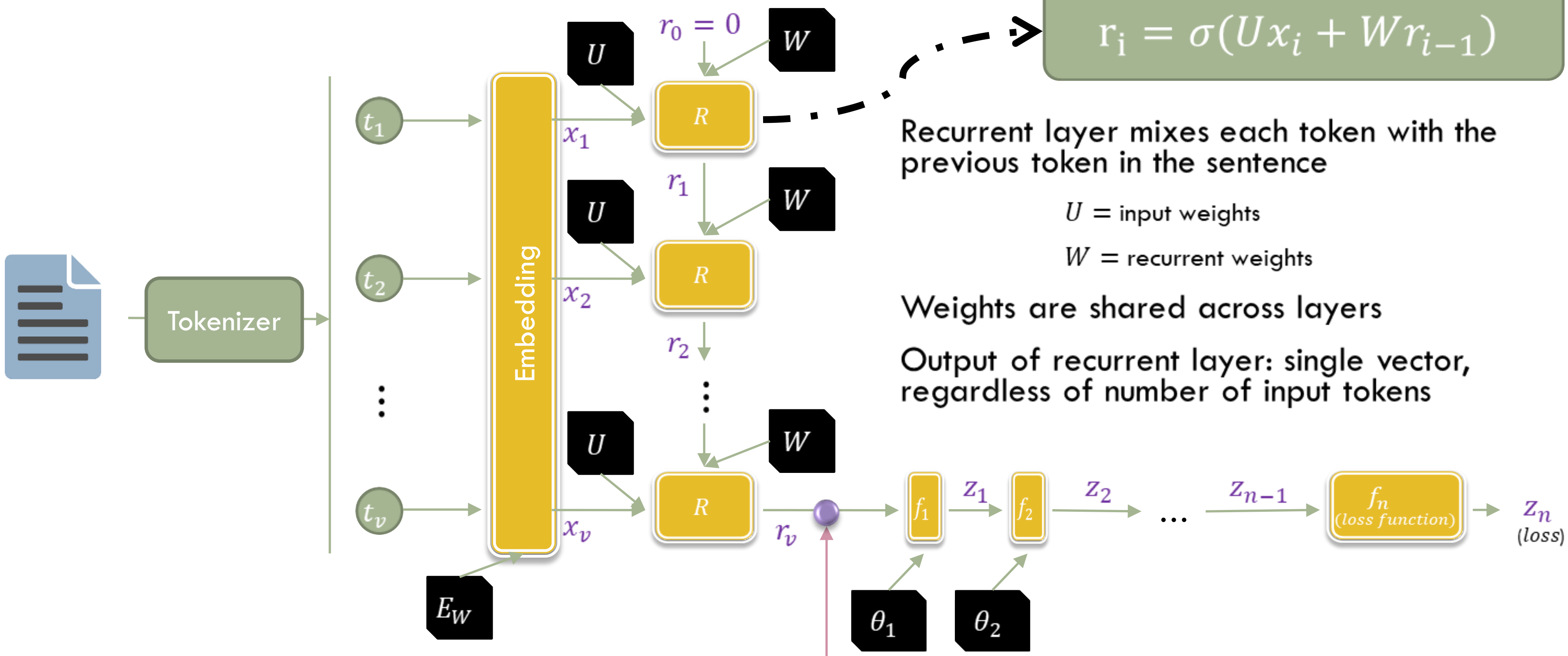


Aplicación: análisis morfosintáctico con ML

- Objetivo: dado un texto tokenizado, generar sus clases morfológicas y árbol de dependencias
- Aproximación para el problema
  - Embeddings
  - Modelos convolucionales de mezcla
  - Capa final de la red que para cada token predice su clase morfológica, padre en el árbol y tipo de relación con el padre
- Recursos
  - spaCy: https://spacy.io/



Better token combinations: recurrent layers



$$\mathrm{r_i} = \sigma(Ux_i + Wr_{i-1})$$

Recurrent layer mixes each token with the previous token in the sentence

U = input weights

W = recurrent weights

Weights are shared across layers

Output of recurrent layer: single vector, regardless of number of input tokens

Fixed d-dimensional representation of whole text that really takes order into account!

Problems when training recurrent layers

Even though recurrent networks have a small number of weights compared to deep networks, because they unfold over each input token they behave like deep networks

In particular, they suffer from severe vanishing gradients!

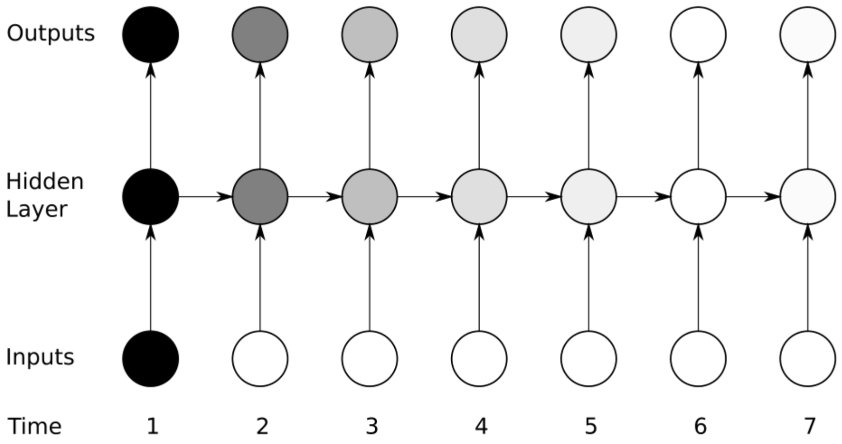


Figure 4.1: **The vanishing gradient problem for RNNs.** The shading of the nodes in the unfolded network indicates their sensitivity to the inputs at time one (the darker the shade, the greater the sensitivity). The sensitivity decays over time as new inputs overwrite the activations of the hidden layer, and the network 'forgets' the first inputs.

Graves - Supervised Sequence Labelling with Recurrent Neural Networks

Dealing with vanishing gradients

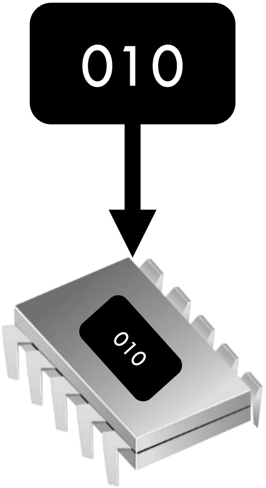
- To deal with vanishing gradients, many approaches have been proposed
  - Smarter weights initialization
  - Rectified Linear Units
  - Non-gradient based methods (e.g. discrete error propagation)

- The most successful aproach, which also improves over the classic RNN design are Long-Short Term Memory (LSTM) Networks

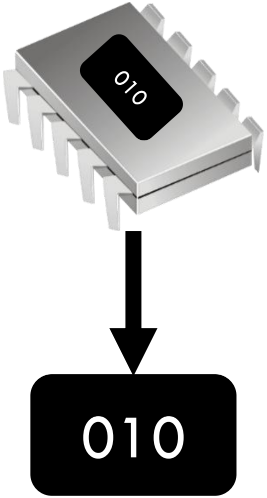
About memories

To understand LSTM units, firts suppose a minimal data-storing device. In order to be usable it requires a minimal set of operations

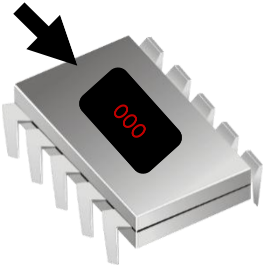
Write



Read



Reset

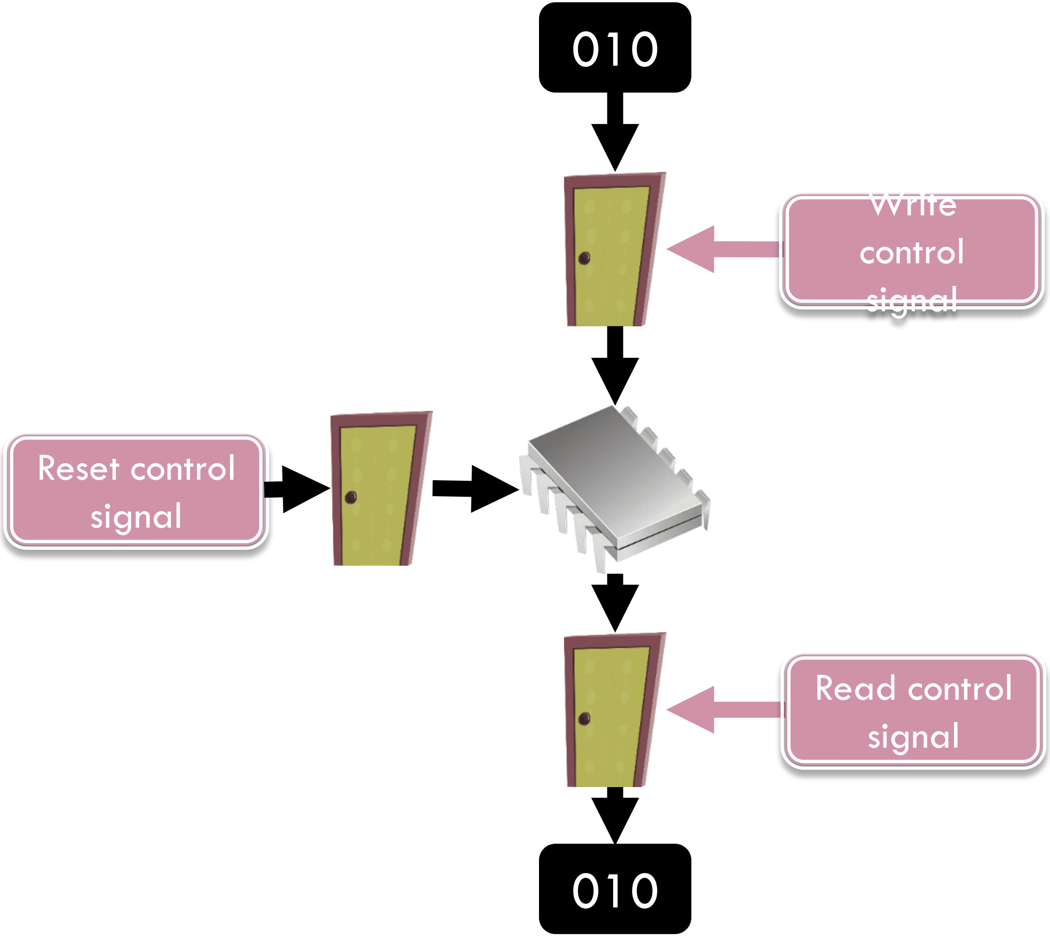


Gated memories

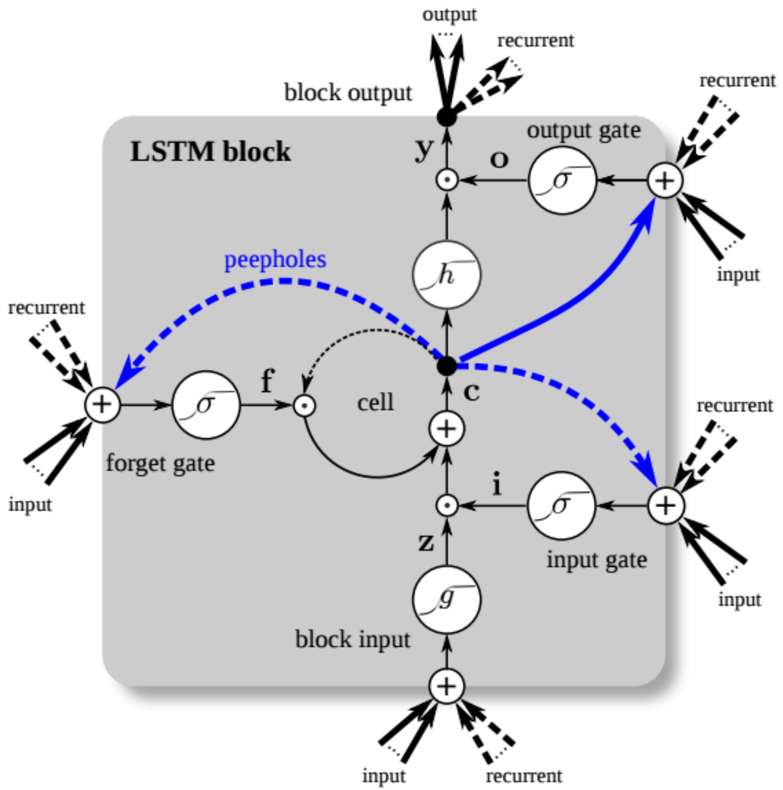
The three operations can be combined, together with some gates that open/close the execution of each operation.

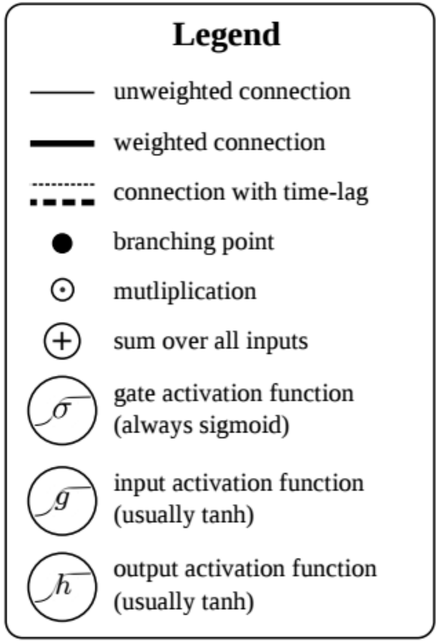
Such gates can regulated through control signals.

This can be implemented into an artificial neuron!



The Long-Short Term Memory unit





Cell: memory storage, keeps a value C_t for future use

Input gate: combines unit inputs to modify the value stored in the cell

Forget gate: attenuates the value stored in the cell

Output gate: combines unit input with cell value to produce the output

All gates receive inputs from previous layer + recurrent inputs from this layer

Deep Learning in a Nutshell: Sequence Learning https://devblogs.nvidia.com/parallelforall/deep-learning-nutshell-sequence-learning

Advantages of LSTMs

Input gates: the network can decide when an input is important enough to be memorized

Reset gates: the network can decide when a memory is no longer useful

Output gates: the network can decide when to release a particular memory to compute the current network output

A memorized value can be retained indefinitely

- Thus, no vanishing gradients!

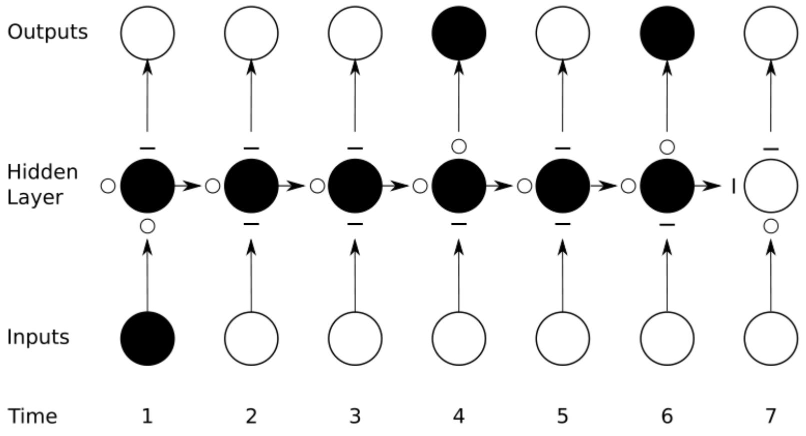
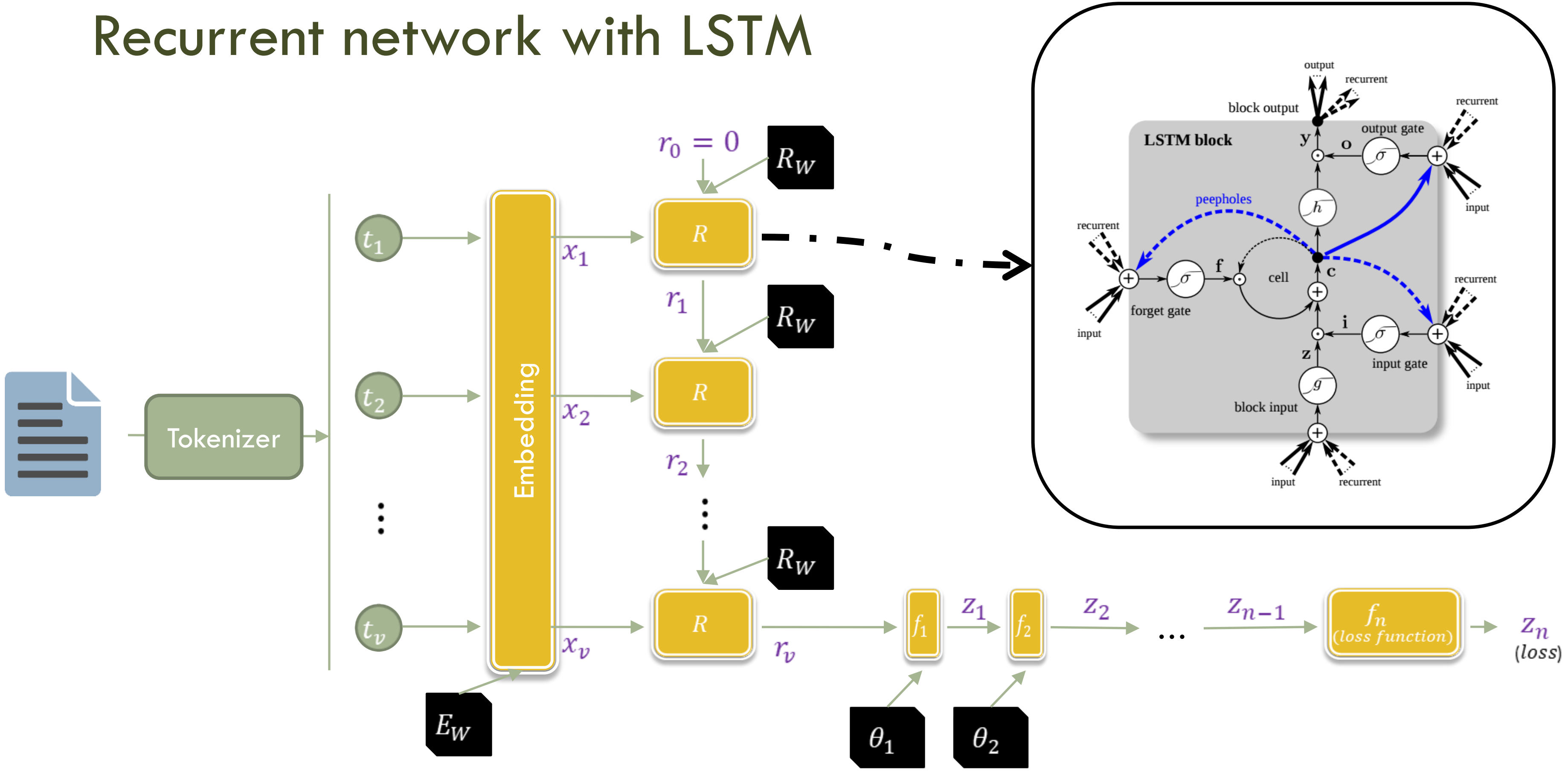


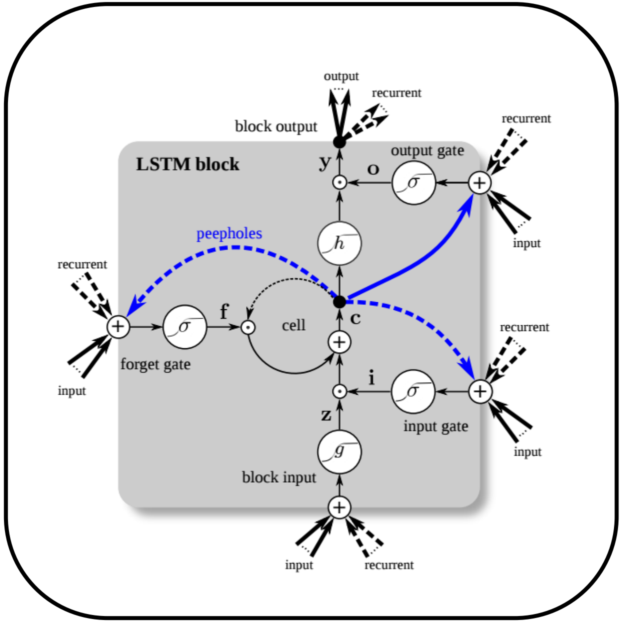
Figure 4.4: **Preservation of gradient information by LSTM.** As in Figure 4.1 the shading of the nodes indicates their sensitivity to the inputs at time one; in this case the black nodes are maximally sensitive and the white nodes are entirely insensitive. The state of the input, forget, and output gates are displayed below, to the left and above the hidden layer respectively. For simplicity, all gates are either entirely open ('O') or closed ('—'). The memory cell 'remembers' the first input as long as the forget gate is open and the input gate is closed. The sensitivity of the output layer can be switched on and off by the output gate without affecting the cell.

Graves - Supervised Sequence Labelling with Recurrent Neural Networks

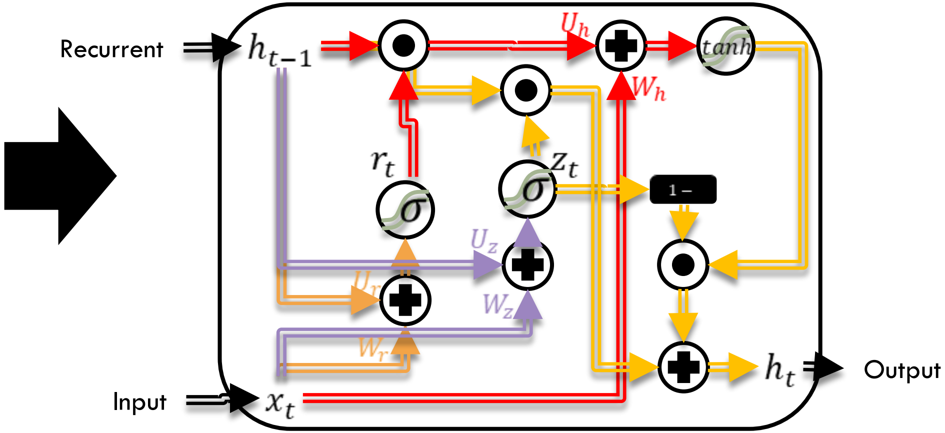
Recurrent network with LSTM



Gated Recurrent Units (GRU)



$$z_t = \sigma(W_z x_t + U_z h_{t-1} + b_z)$$
$$r_t = \sigma(W_r x_t + U_r h_{t-1} + b_r)$$
$$h_t = z_t \circ h_{t-1} + (1 - z_t) \circ tanh(W_h x_t + U_h (r_t \circ h_{t-1}) + b_h)$$

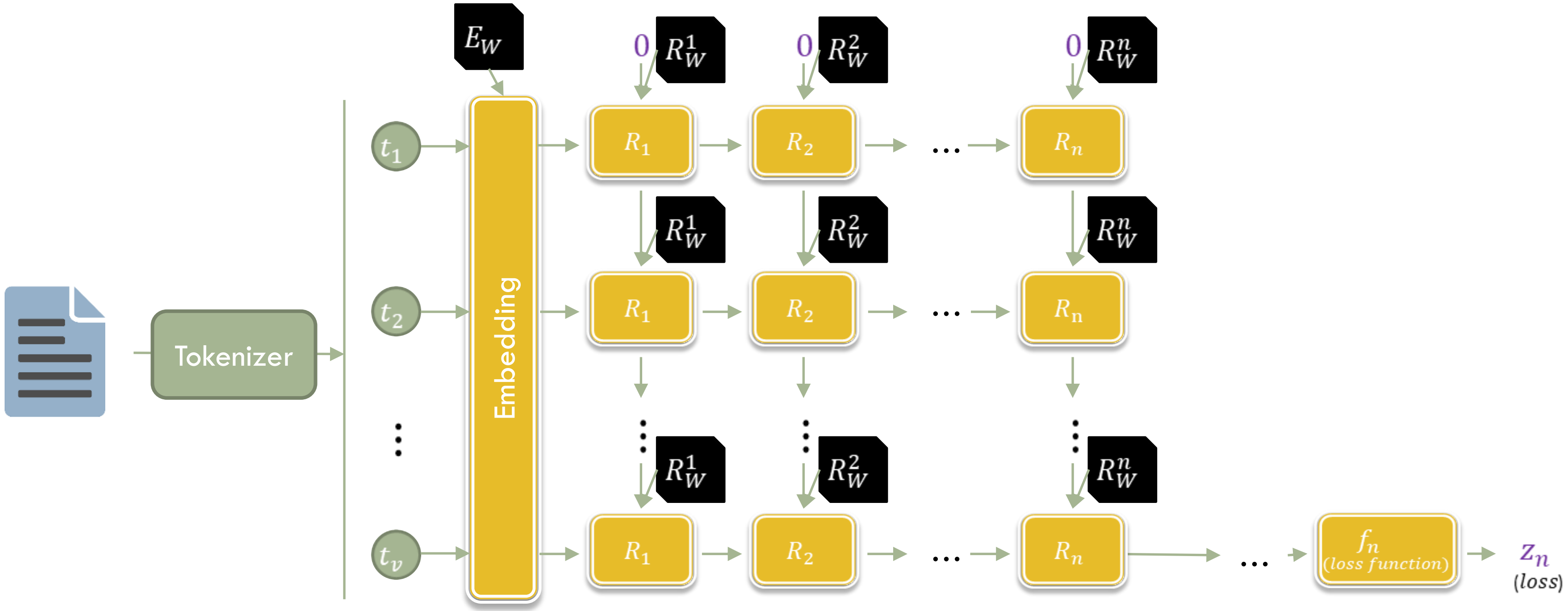


- Simplified version of the LSTM cell
  - Combine "write" and "reset" gates into a single "update" gate. No "read" gate: output values h_t are the memory values themselves
- Lower computational cost
- Works better than LSTM in smaller datasets

Cho et al - Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation

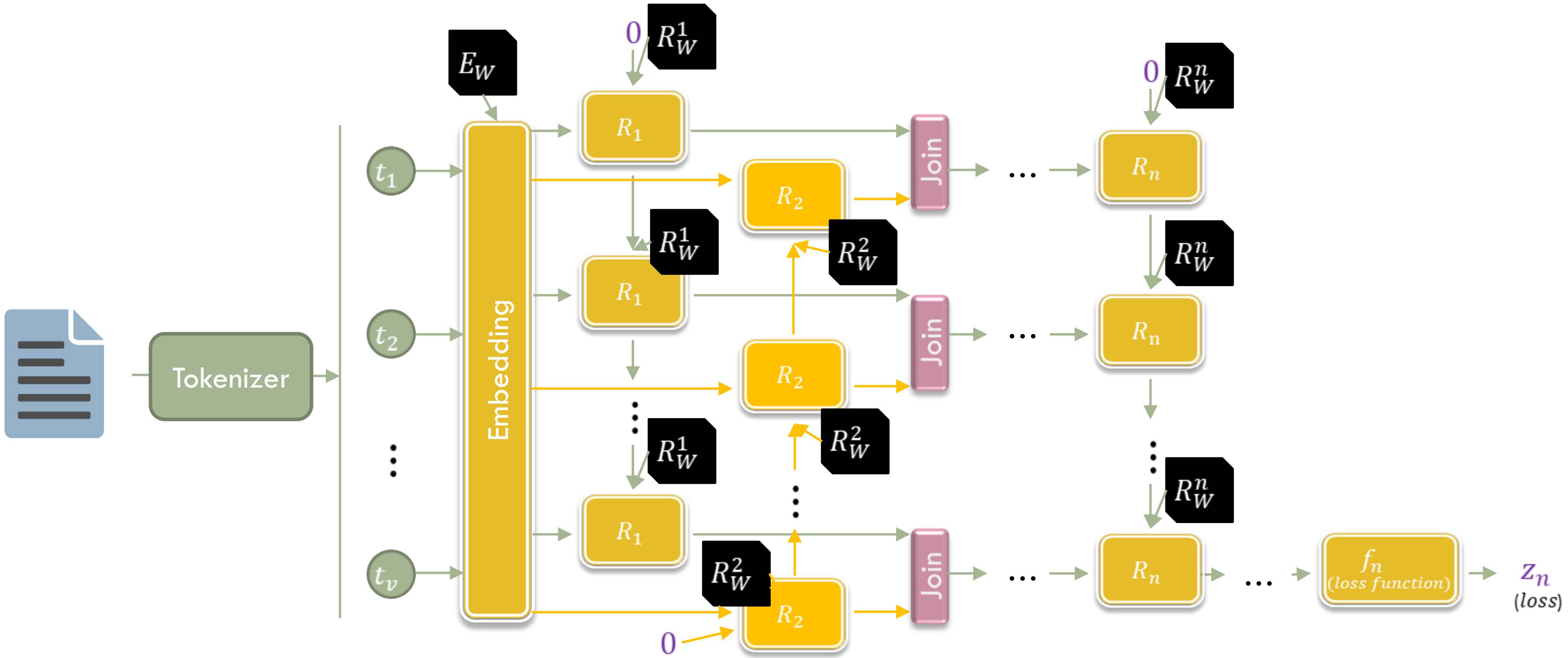
Colah – Understanding LSTM Networks - http://colah.github.io/posts/2015-08-Understanding-LSTMs/

Stacked LSTM



- As with other kind of layers, LSTM layers can be stacked on top of one another
- Each LSTM layer mixes the outputs of the previous layer
- Refined mixings of the observed sequences

Bidirectional LSTM (BiLSTM)



- Follow the Embedding with two LSTM layers, reading data in opposite directions
- Join the outputs of both layers (concat, mean), and propagate to next layers
- Better mixings of the observed sequences

Example: simple language generation

Train a 3-layered LSTM to predict the next character in a given text

Predict using the previous 100 characters of the text

Then generate new texts from the network!

Shakespeare

PANDARUS:

Alas, I think he shall be come approached and the day

When little srain would be attain'd into being never fed,

And who is but a chain and subjects of his death,

I should not sleep.

Second Senator:

They are away this miseries, produced upon my soul,

Breaking and strongly should be buried, when I perish

The earth and thoughts of many states.

Lovecraft

his squalid mother and grandfather in this respect was thought very notable until the horror of 1980, but the old tombs and sense of stone of the one had a hard connection which he had been there where the present horror started at the cry of the last the room of the sea. The neighbourhood of the hall it was to meet a catch of marine many of the room seemed to be indeed to see a marvellous ground of a trace of of the space we can see their specimens; and so the death of the sea some of the…

Movie Titles

Better Story<END>

Last Company<END>

The Love Balls: Part 2<END>

The Salence Truth of Boys<END>

Really Case to Disaster<END>

Ana House Thief<END>

The Secret of the Cast<END>

The Countdust of Story<END>

We We Travele<END>

The Tale of the Trome<END>

The Vecyme That White Edition<END>

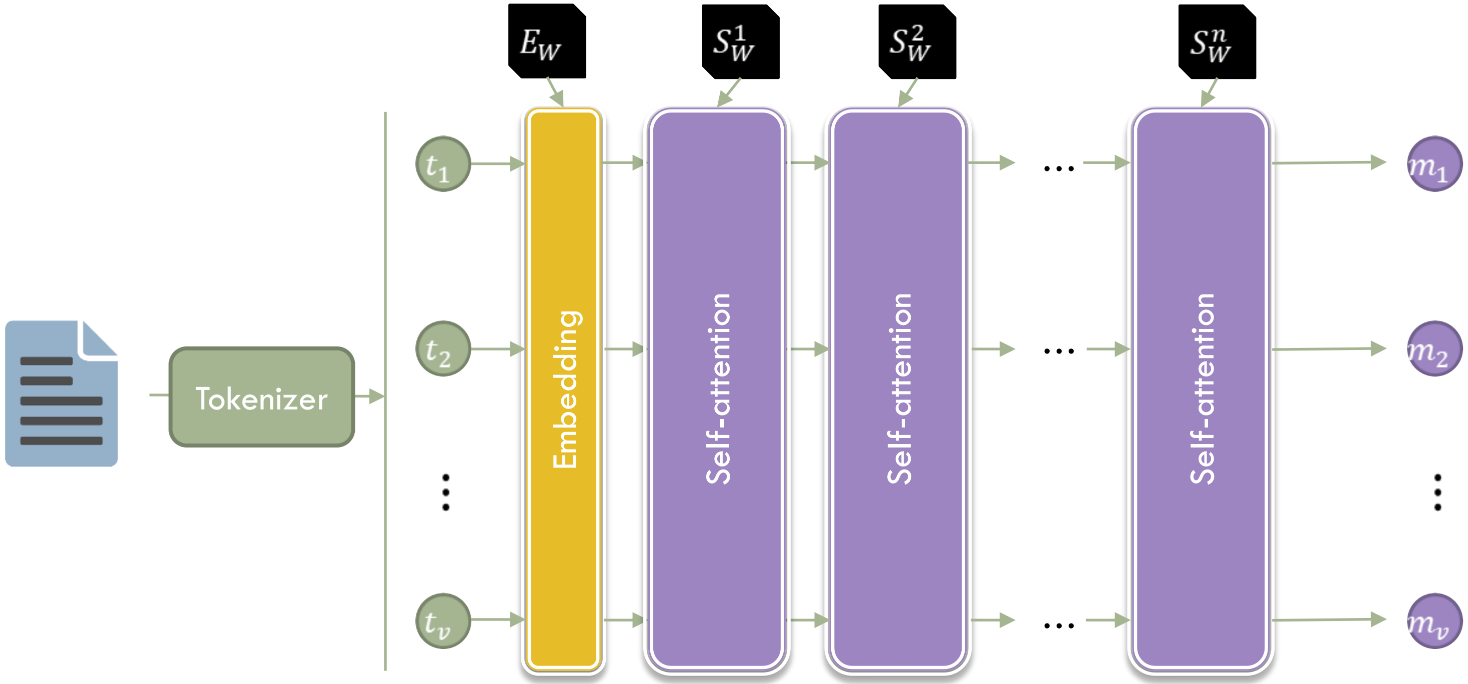
All Bedroom<END>

Alive a Fall<END>

Karpathy - The Unreasonable Effectiveness of Recurrent Neural Networks http://karpathy.github.io/2015/05/21/rnn-effectiveness/

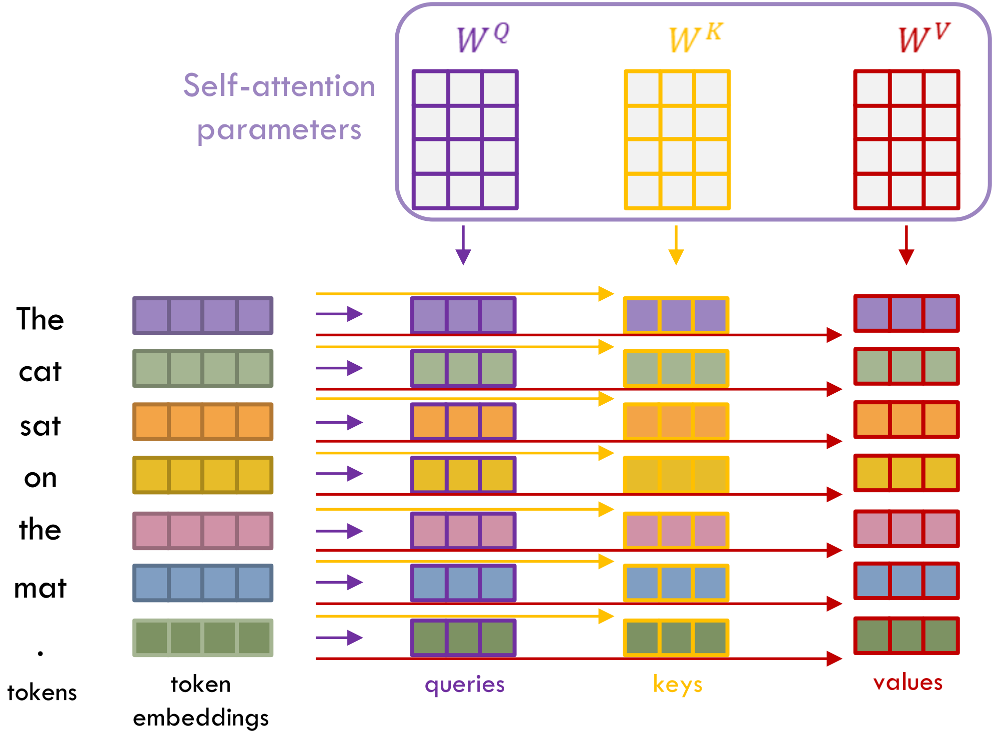
Neurowriter - https://github.com/albarji/neurowriter

Self-attention



- Improving over recurrent networks, the self-attention layer mixes the representation of every token in the document with every other token, and produces a new embedding for each token.
- Contextualized embeddings.

Self-attention in detail



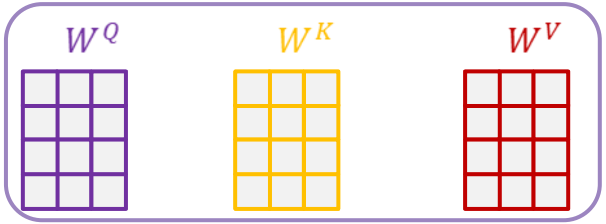
- Multiply each token embedding by matrices W^Q, W^K and W^V to produce a query, key and value vector for each token.
- Query vectors: what this token is "looking for" in other tokens
- Key vectors: what this token "can offer" to other tokens
- Relevance score: how aligned is a query vector for a token with the key vector of another token ($q_i \cdot k_j$)
  - High relevance scores mean this pair of tokens must be mixed.
- New embeddings will be produced by mixing value vectors weighted by scores.

Jay Allamar – The Illustrated Transformer - https://jalammar.github.io/illustrated-transformer/

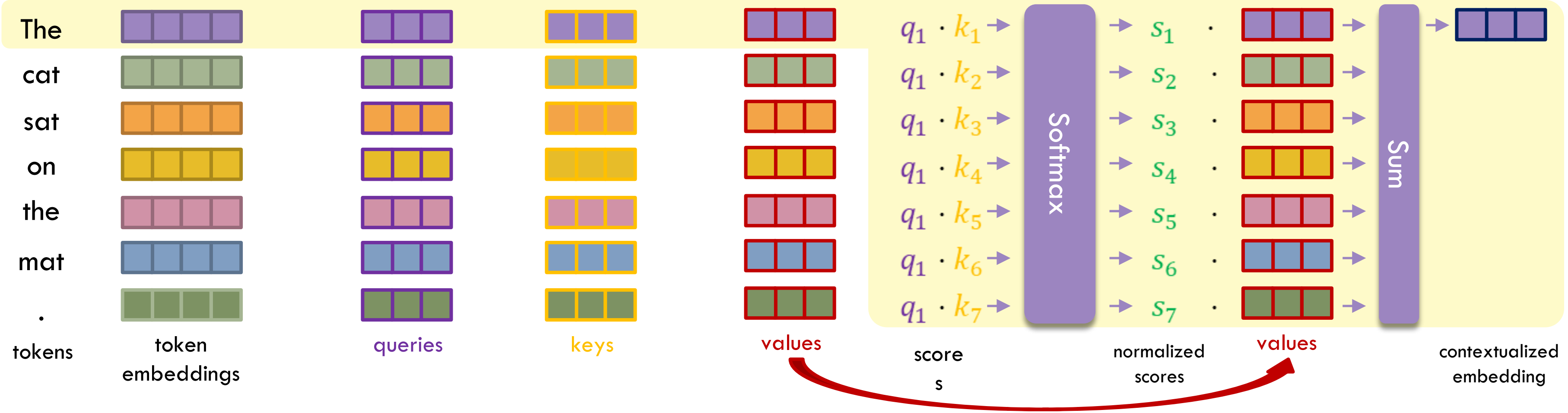
Vaswani et al – Attention is All You need

Self-attention scores for a token

Self-attention parameters



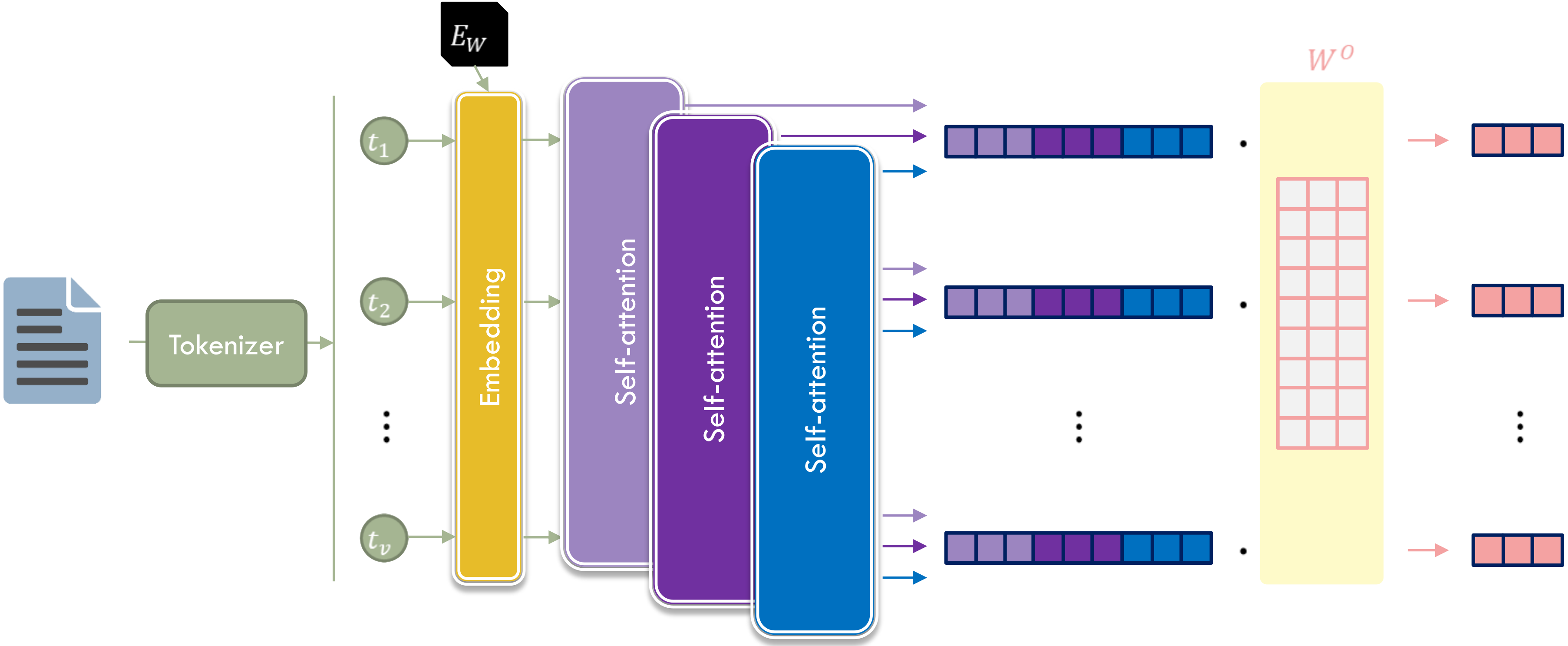
- The contextualized embedding for each token mixes value vectors from all words, weighted by query · key scores.
- Flow for first token:



Jay Allamar – The Illustrated Transformer - https://jalammar.github.io/illustrated-transformer/

Vaswani et al – Attention is All You need

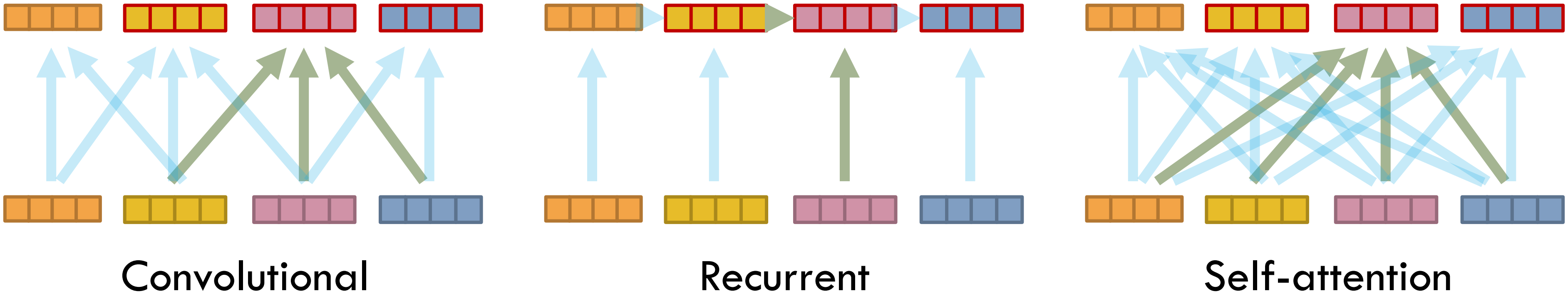
Multi-headed self-attention



- Multiple self-attention blocks can be used in parallel, similarly to multiple convolutional kernels.
- For n blocks (heads) n contextualized embeddings are obtained for each token.
- Concatenate embeddings and mix using an output matrix W^O to recover a single embedding per token.

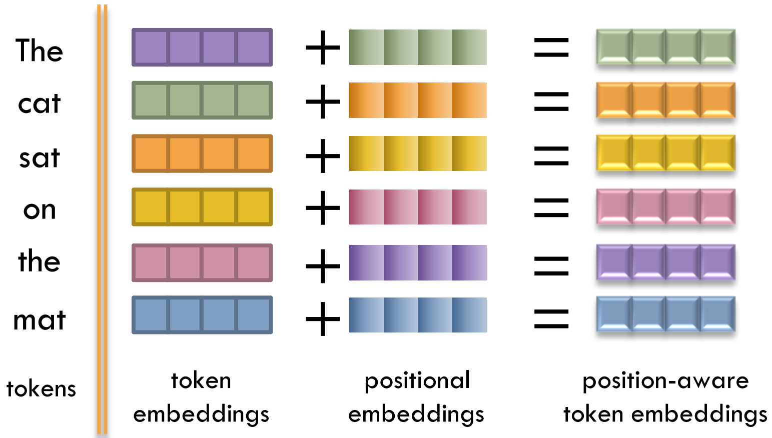
Summary of mixing models

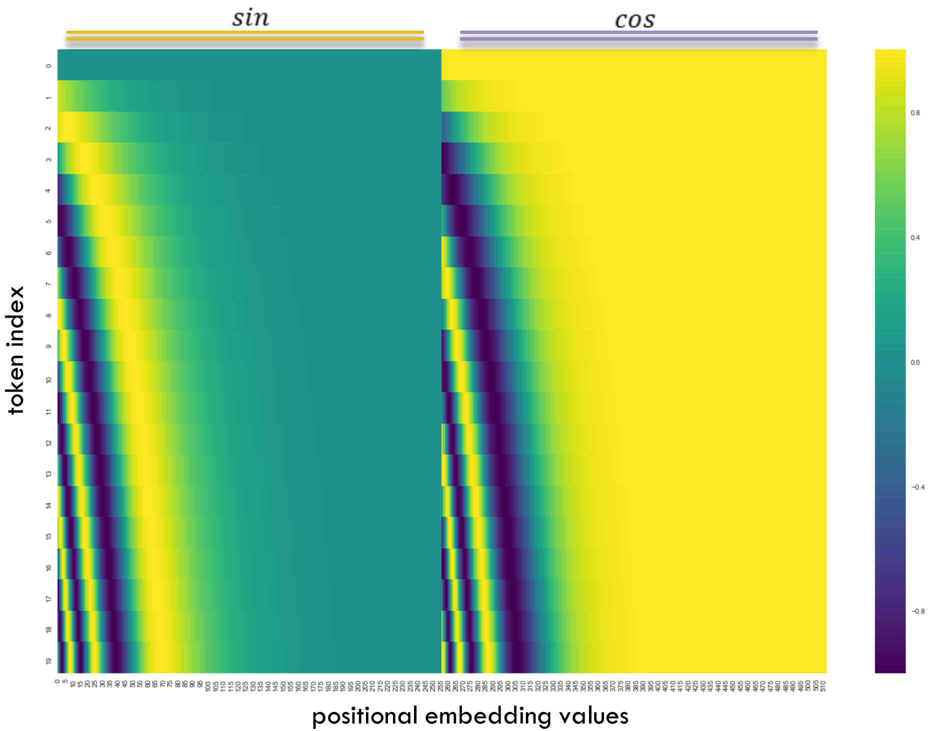
Mixing model	Operation	Sensitive to tokens relative ordering?	Sensitive to tokens absolute position in the document?
Convolutional	Mix each token with neighbouring tokens	✓	✗
Recurrent	Mix each token with new representation of previous token	✓	✓
Self-attention	Mix each token with every other token	✗	✗



Positional embeddings

Self-attention and convolutional mixings can be made aware of tokens relative and absolute positions by introducing positional embeddings.





Positional embeddings values are computed as sin and cos of the position of the word in the document. Each embedding dimension i accounts for a sinusoid of a different frequency, normalized by maximum document length. Positional embeddings have the same size d_{model} has token embeddings.

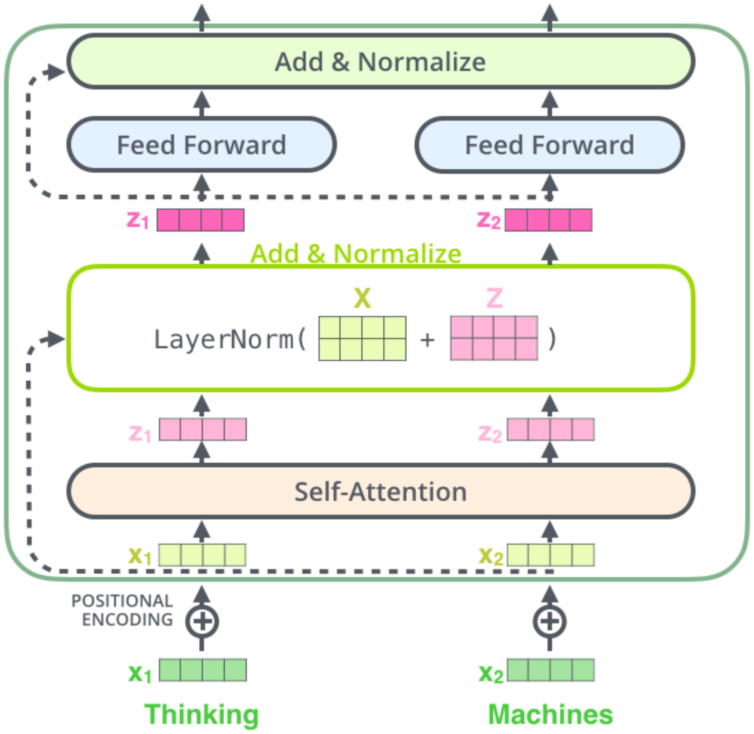
$$PE_{(pos,2i)} = \sin\left(\frac{pos}{maxlen^{\left(\frac{2i}{d_{model}}\right)}}\right) \quad PE_{(pos,2i+1)} = \cos\left(\frac{pos}{maxlen^{\left(\frac{2i}{d_{model}}\right)}}\right)$$

Jay Allamar – The Illustrated Transformer - https://jalammar.github.io/illustrated-transformer/

Vaswani et al – Attention is All You need

The Transformer

An architecture combining all the topics covered so far is the Transformer, which produces state-of-the-art results in complex tasks such as language translation. A Transformer stacks several blocks in the form:



- Token + positional embeddings (in first block)
- Multi-head self-attention
- Residual block with Layer Normalization
- A small Feed Forward network: Dense + ReLU + Dense. This network is applied at each position, separately and identically.
    - This is equivalent to Conv + ReLU + Conv with kernel size 1
- Residual block with Layer Normalization

Jay Allamar – The Illustrated Transformer - https://jalammar.github.io/illustrated-transformer/

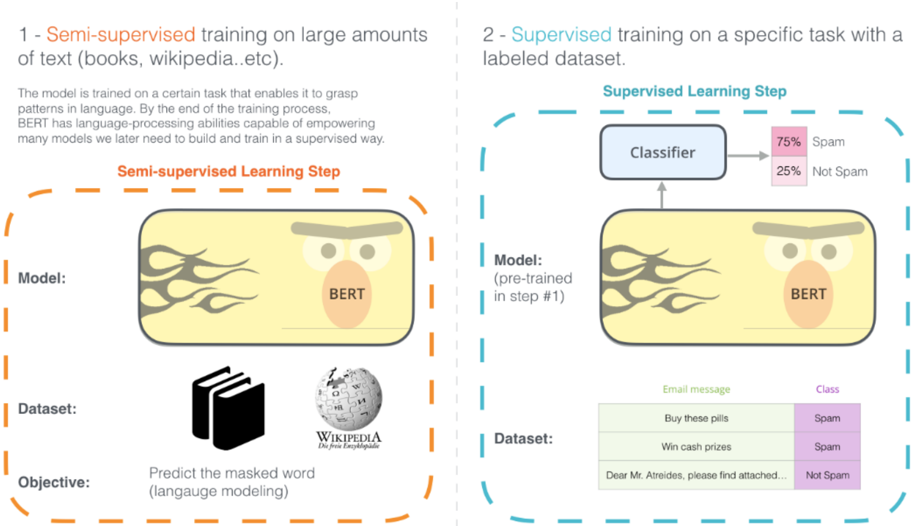
Vaswani et al – Attention is All You need

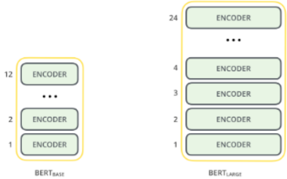
Ba et Al – Layer Normalization

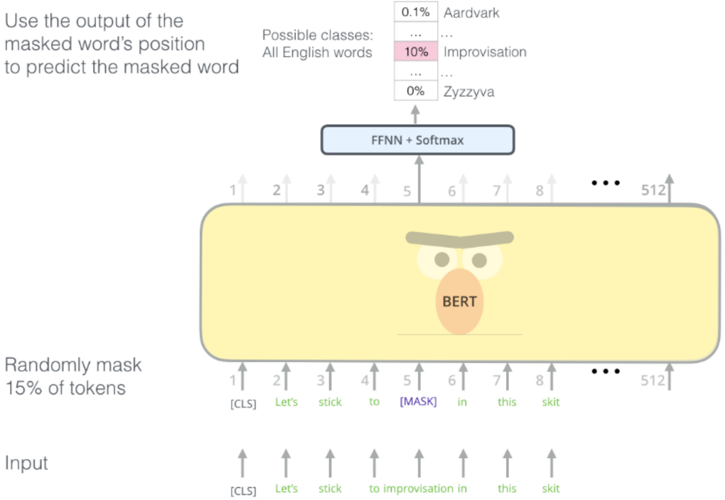
Large Language Models

Large Language models: BERT

It is possible to train very large language models that provide us with not only better token representations, but with a whole stack of layers able to produce contextualized embeddings. The Bidirectional Encoder Representations for Transformers (BERT) model is a stack of Transformers that does this:







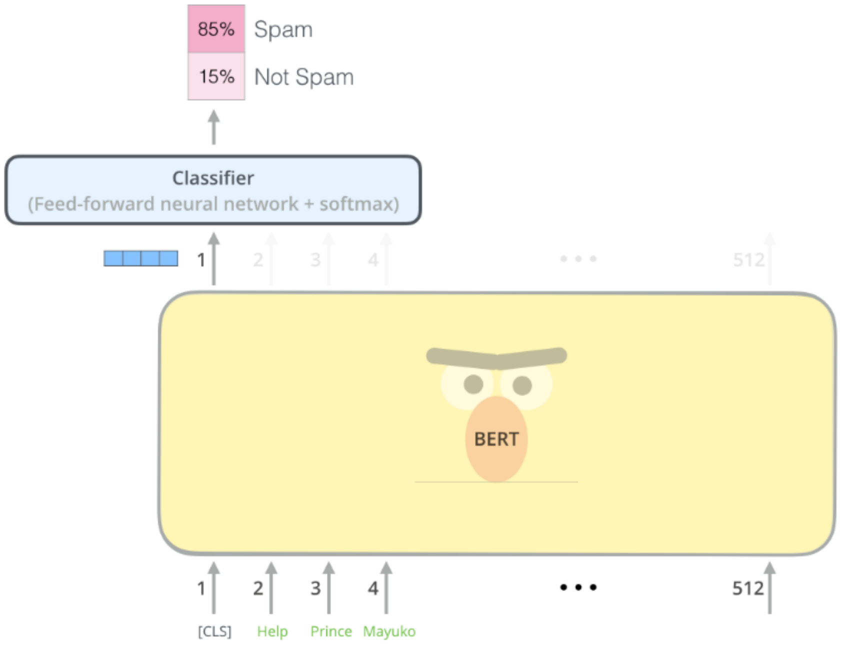
Jay Allamar – The Illustrated BERT, ELMo an co - https://jalammar.github.io/illustrated-bert/

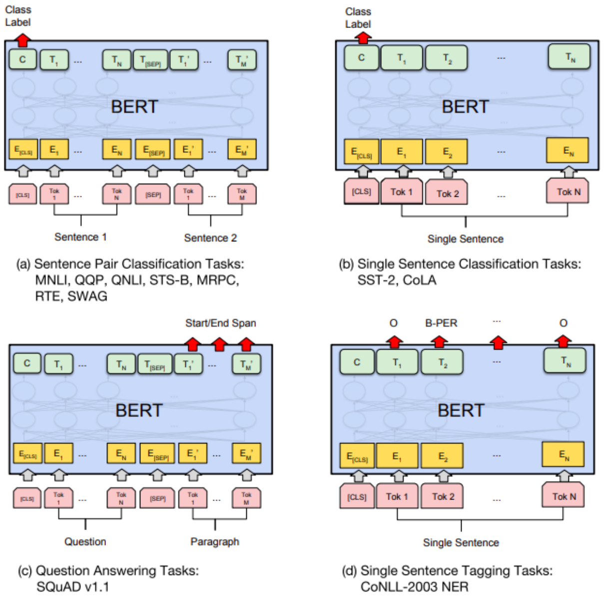
Devlin et al – BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding

BERT pre-trained models: https://github.com/google-research/bert

Large Language models: BERT as transfer learning

BERT can be used as a transfer learning tool, to obtain better embeddings for our documents. We can then train a small network on BERT outputs for our particular problem.





Jay Allamar – The Illustrated BERT, ELMo an co - https://jalammar.github.io/illustrated-bert/

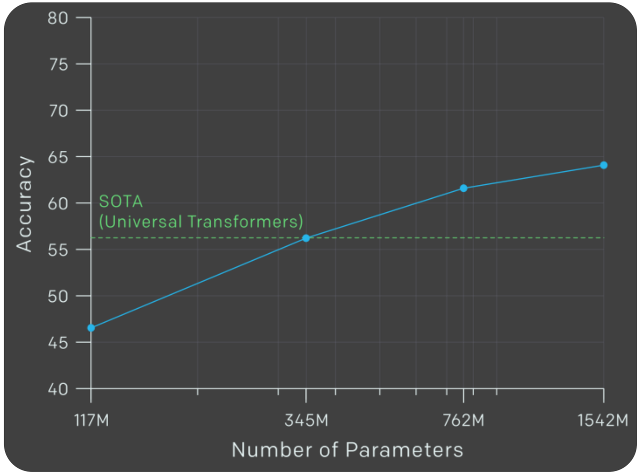
Devlin et al – BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding

BERT pre-trained models: https://github.com/google-research/bert

Large Language models: GPT-2

DATASET	METRIC	OUR RESULT	PREVIOUS RECORD	HUMAN
Winograd Schema Challenge	accuracy (+)	**70.70%**	63.7%	92%+
LAMBADA	accuracy (+)	**63.24%**	59.23%	95%+
LAMBADA	perplexity (-)	**8.6**	99	~1-2
Children's Book Test Common Nouns (validation accuracy)	accuracy (+)	**93.30%**	85.7%	96%
Children's Book Test Named Entities (validation accuracy)	accuracy (+)	**89.05%**	82.3%	92%
Penn Tree Bank	perplexity (-)	**35.76**	46.54	unknown
WikiText-2	perplexity (-)	**18.34**	39.14	unknown
enwik8	bits per character (-)	**0.93**	0.99	unknown
text8	bits per character (-)	**0.98**	1.08	unknown
WikiText-103	perplexity (-)	**17.48**	18.3	unknown

GPT-2 is an even larger Transformer model, up to 48 layers, trained on Internet texts extracted from Reddit. Produces a very accurate language model.



Performance in LAMBADA language modelling task

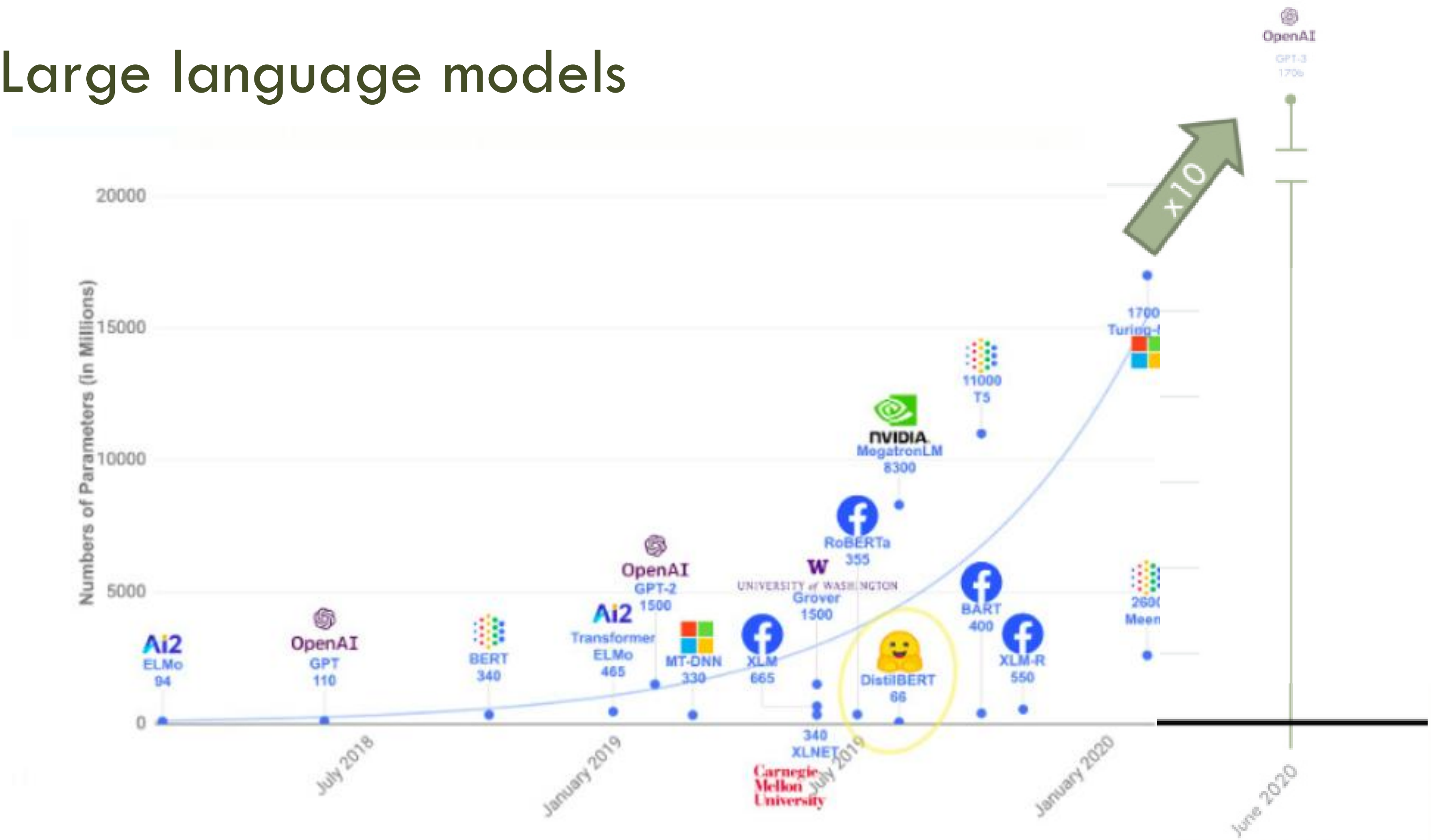
	Parameters	Layers	d_{model}
	117M	12	768
BERT Large	345M	24	1024
	762M	36	1280
	1542M	48	1600

Table 2. Architecture hyperparameters for the 4 model sizes.

OpenAI - Better Language Models and their implications - https://blog.openai.com/better-language-models

Racford et al – Language Models are Unsupervised Multitask Learners

Large language models

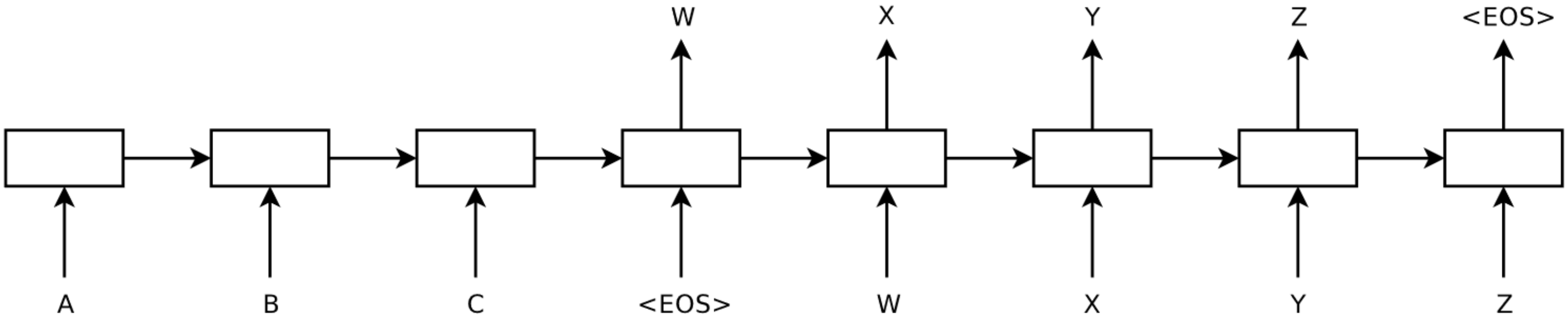


NLP models through time, with their number of parameters

https://research.aimultiple.com/gpt/

Sequence to Sequence Learning

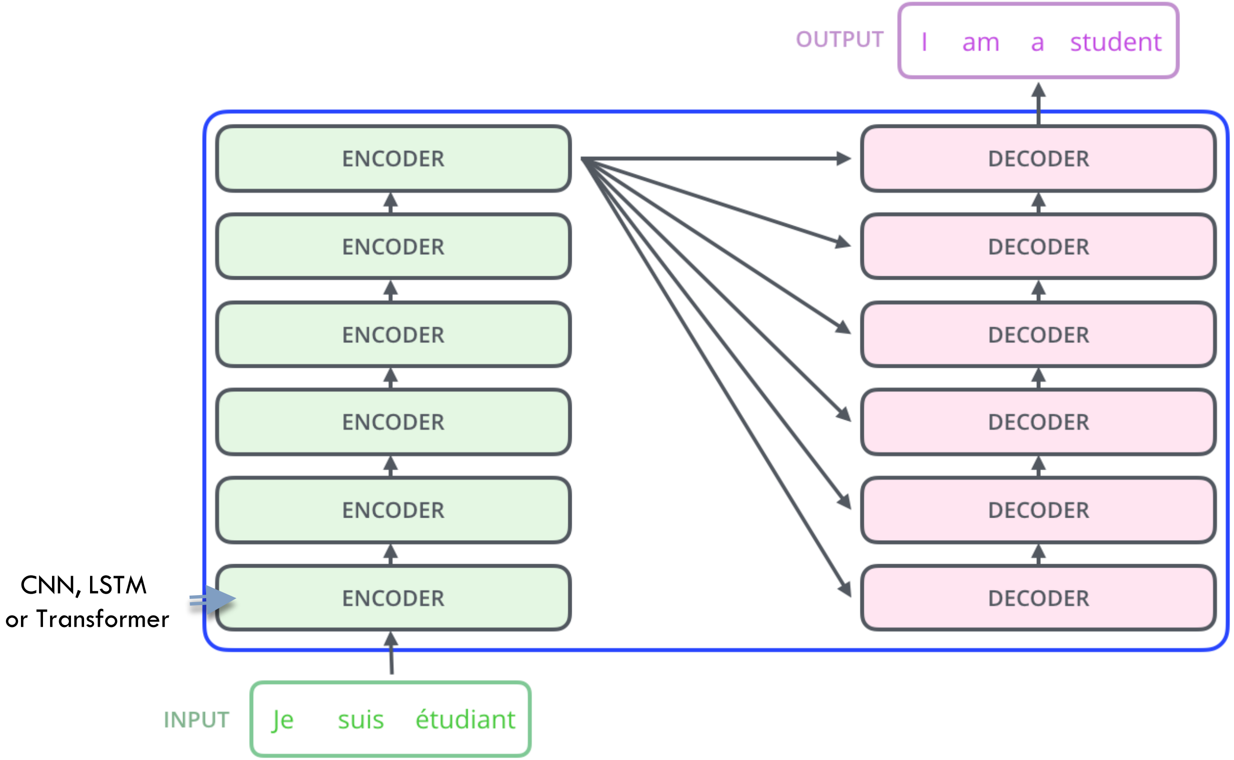
Sequence to sequence learning



- The model reads the input sequence A + B + C and produces W + X + Y + Z as output
- After an special input <EOS> is received the hidden state of the network is used as input to the decoder network
- The decoder network generates one sequence element at a time, which is fed back as input to produce the next element
- The decoder stops once the special character <EOS> is generated

Sutskever et al: Sequence to Sequence Learning with Neural Networks

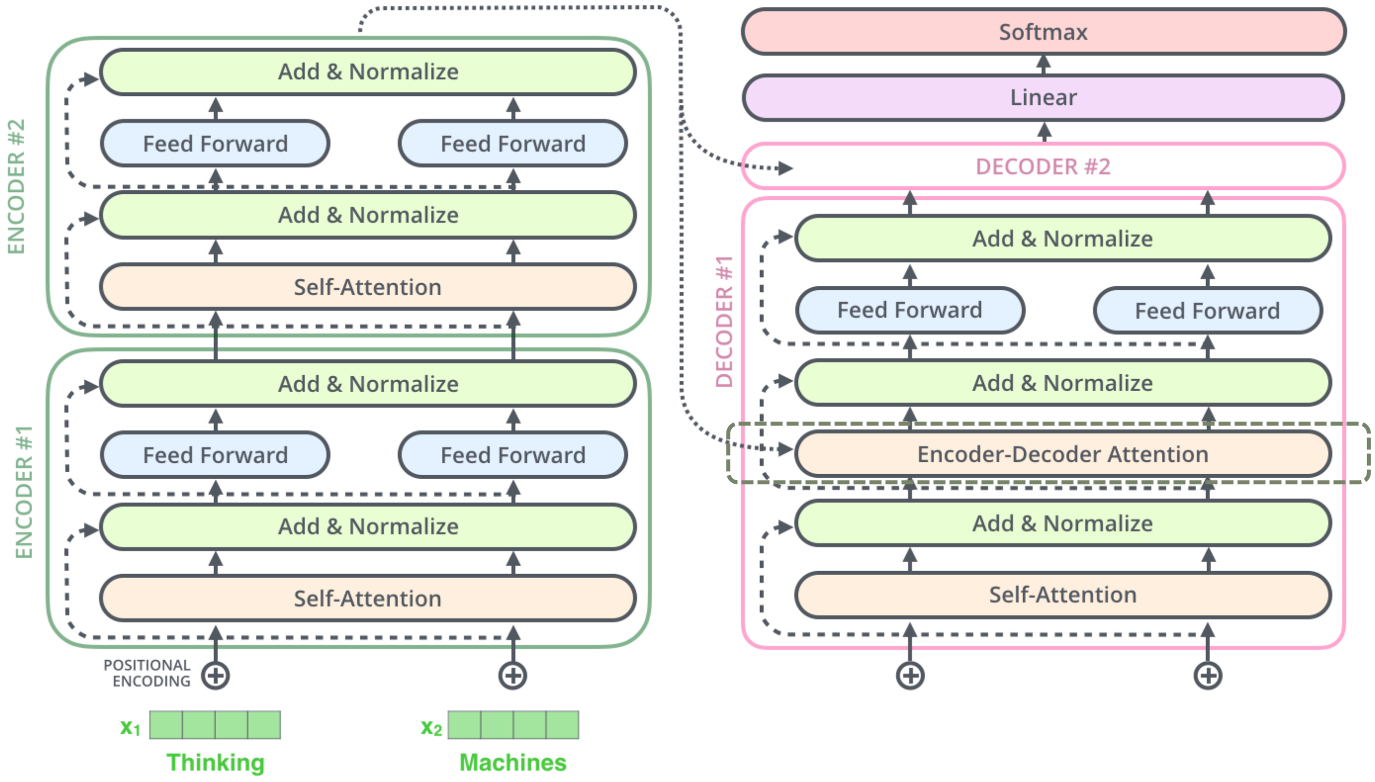
Example: Machine Translation



Jay Allamar – The Illustrated Transformer - https://jalammar.github.io/illustrated-transformer/

Vaswani et al – Attention is All You need

Encoder-Decoder architecture with Transformers



Mix encoder outputs with translation so far

Jay Allamar – The Illustrated Transformer - https://jalammar.github.io/illustrated-transformer/

Vaswani et al – Attention is All You need

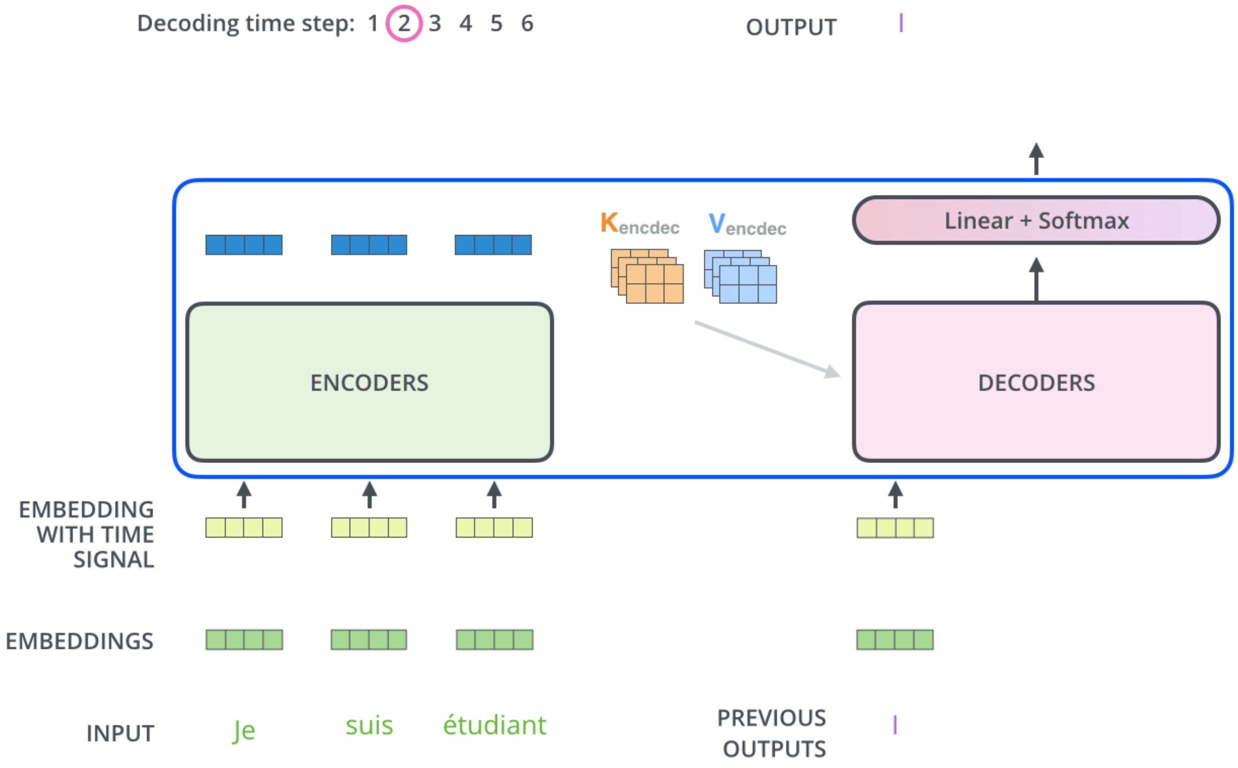
Encoder-Decoder architecture: first decoding step

No se puede mostrar la imagen. Puede que su equipo no tenga suficiente memoria para abrir la imagen o que ésta esté dañada. Reinicie el equipo y, a continuación, abra el archivo de nuevo. Si sigue apareciendo la x roja, puede que tenga que borrar la imagen e insertarla de nuevo.

Jay Allamar – The Illustrated Transformer - https://jalammar.github.io/illustrated-transformer/

Vaswani et al – Attention is All You need

Encoder-Decoder architecture: next decoding steps



Jay Allamar – The Illustrated Transformer - https://jalammar.github.io/illustrated-transformer/

Vaswani et al – Attention is All You need

Aplicación: traducción automática

- **Objetivo**: dado un texto en un idioma conocido, traducirlo a otro idioma dado
- **Aproximación** para el problema
  - Embeddings, Transformers, modelos grandes de lenguaje (tipo BERT)
  - Modelo sequence-to-sequence que pase de la secuencia de un idioma a otro
- **Recursos**
  - Corpus alineado del parlamento Europeo: http://www.statmt.org/europarl/

Language translation is one of the ways we can give people the power to build community and bring the world closer together. It can help people connect with family members who live overseas, or better understand the perspective of someone who speaks a different language. We use machine translation to translate text in posts and comments automatically, in order to break language barriers and allow people around the world to communicate with each other.

https://code.facebook.com/posts/289921871474277/transitioning-entirely-to-neural-machine-translation/

Aplicación: respuesta de preguntas sobre texto

- **Objetivo**: dado un texto que presente detalles sobre un tema concreto, y una pregunta sobre ese tema, marcar en el texto dónde está la respuesta a la pregunta.
- **Aproximación** para el problema
  - Modelos grandes de lenguaje (tipo BERT)
  - Modelo sequence-to-sequence que reciba el texto de la pregunta y del texto, y para cada token del texto indique si forma parte de la respuesta o no
- **Recursos**
  - The Stanford Question Answering Dataset https://rajpurkar.github.io/SQuAD-explorer/

SQuAD2.0

The Stanford Question Answering Dataset

European Union law is a body of treaties and legislation, such as Regulations and Directives, which have direct effect or indirect effect on the laws of European Union member states. The three sources of European Union law are primary law, secondary law and supplementary law. The main sources of primary law are the Treaties establishing the European Union. Secondary sources include regulations and directives which are based on the Treaties. The legislature of the European Union is principally composed of the European Parliament and the Council of the European Union, which under the Treaties may establish secondary law to pursue the objective set out in the Treaties.

What is European Union Law?

Ground Truth Answers: a body of treaties and legislation | a body of treaties and legislation, such as Regulations and Directives, which have direct effect or indirect effect on the laws of European Union member states | a body of treaties and legislation, such as Regulations and Directives | a body of treaties and legislation, such as Regulations and Directives

Prediction: a body of treaties and legislation