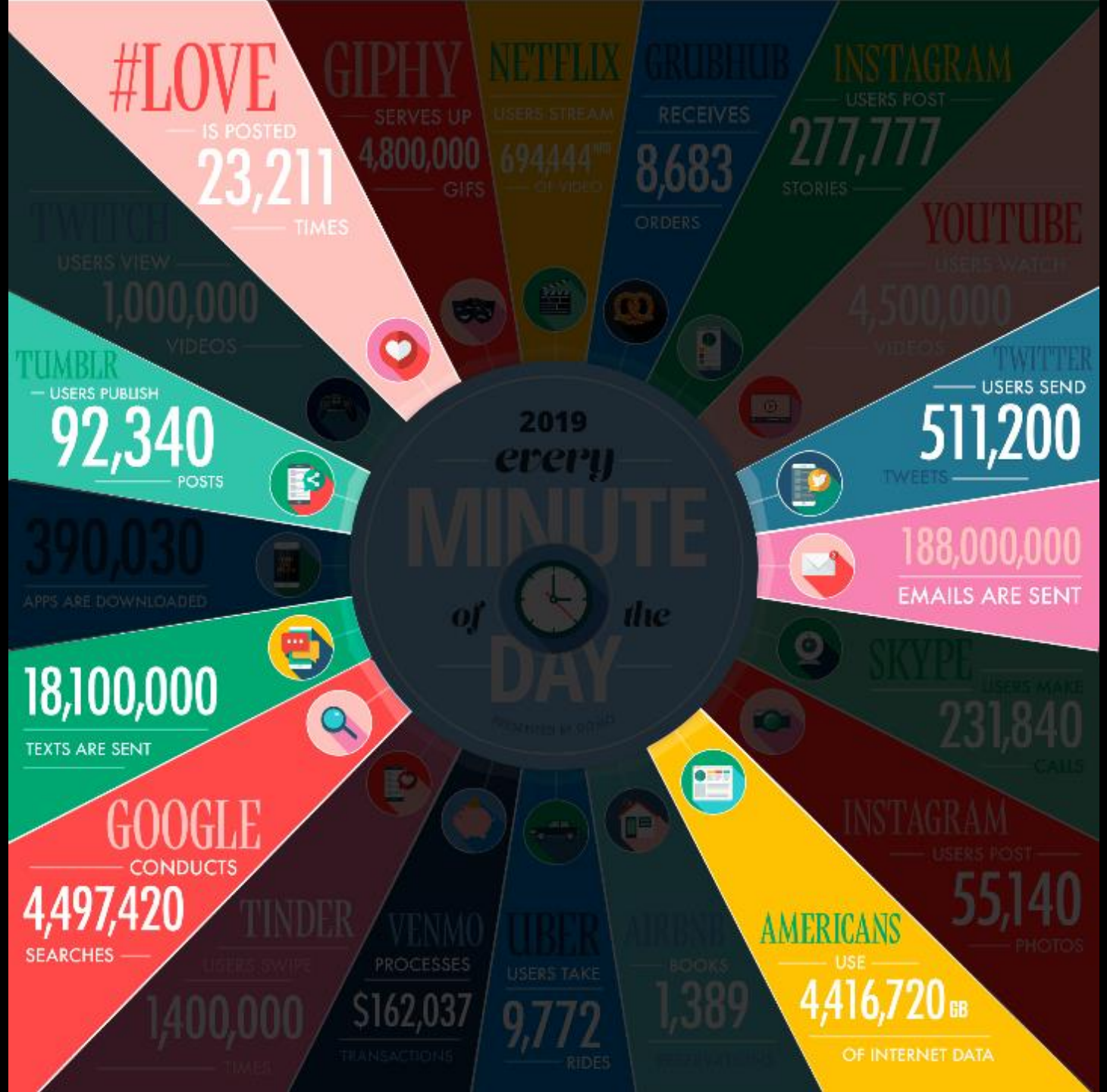
Deep Learning for text

Álvaro Barbero Jiménez
alvaro.barbero.jimenez@gmail.com



Infographic by https://www.digitalinformationworld.com/2019/07/data-never-sleeps-7-infographic.html

Characters

Language minimal unit

国 k あ á i

A α u ℴ ö ñ

Phonemes

Units of sound

/a/ /k/

/č/ /ñ/

Words or tokens

Groups of several characters

España

England

日本

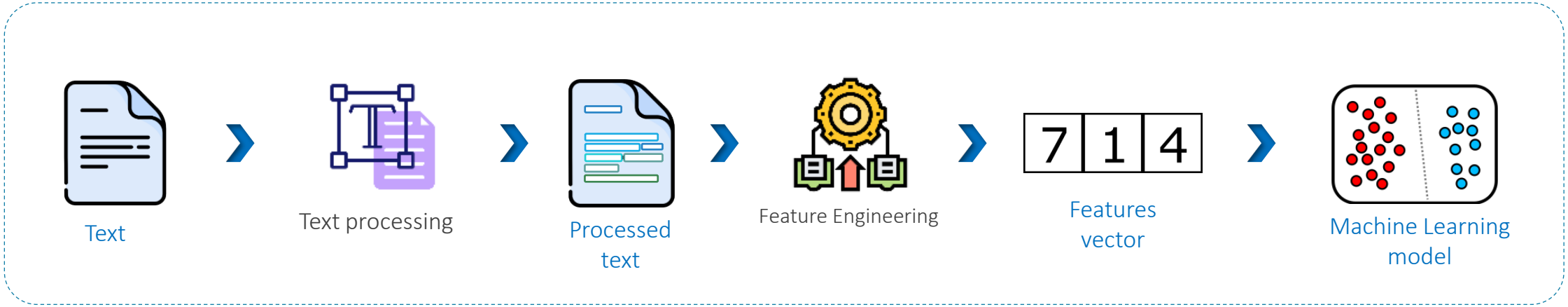
Sentences

Groups of words

“The quick brown fox jumps over the lazy dog.”

“El veloz murciélago hindú comía cardillo y kiwi.”

Classic text processing pipeline



- Do some processing to the text to remove useless pieces, split tokens, normalize words…
- Compute representative features from the processed text to produced fixed-length vectors.
- Train a Machine Learning model with the features vectors.

Basic text processing: tokenization

Group raw characters into meaningful tokens (usually words)

"The quick brown fox jumps over the lazy dog."

["The", "quick", "brown", "fox", "jumps", "over", "the", "lazy", "dog", "."]

Non-trivial for some languages!

昨日、吉森くんはクラスでいませんでした。病気になったかもしれません。。。ちょっと心配ですよ

It's harder than it seems! Trust the professionals

Natural Language ToolKit

python™

spaCy

FreeLing

Stanza

NLTK

spaCy

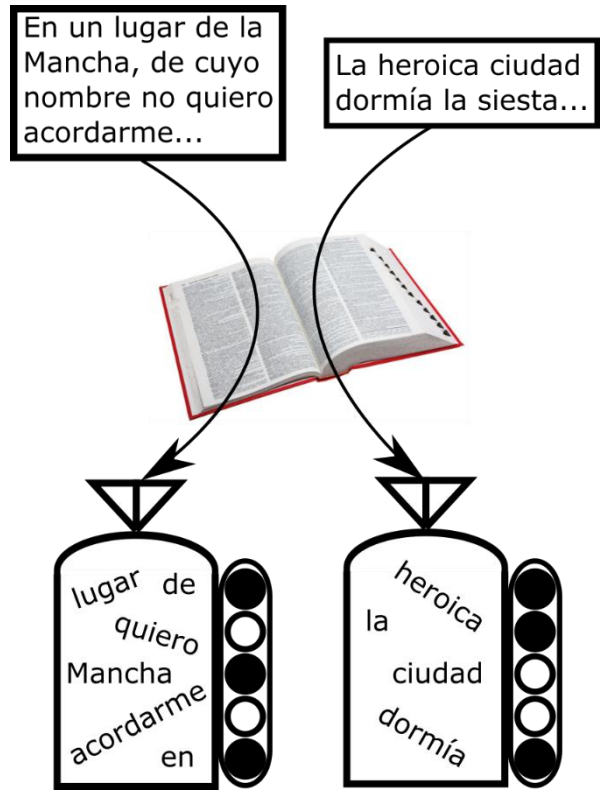
Freeling

Stanza

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

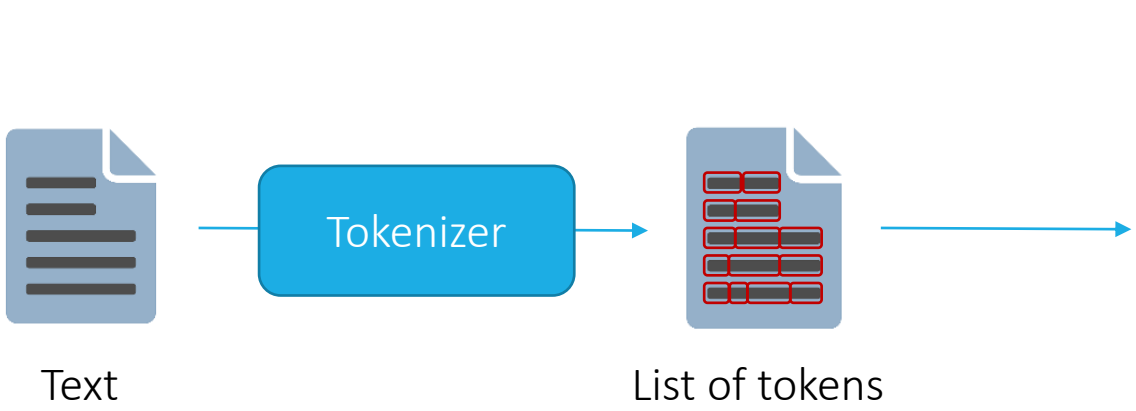
- Bag of words one-hot vectors work, but lack semantic meaning
- All words are equally apart in vector space, regardless of their meaning (semantics) or function (syntax)

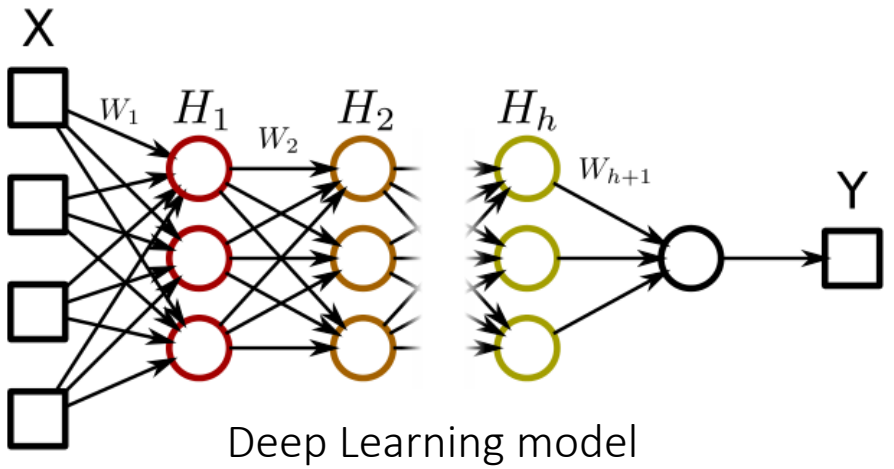
- Are there better encodings for words?
- Even more: can we learn them automatically?

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!





Deep Learning model

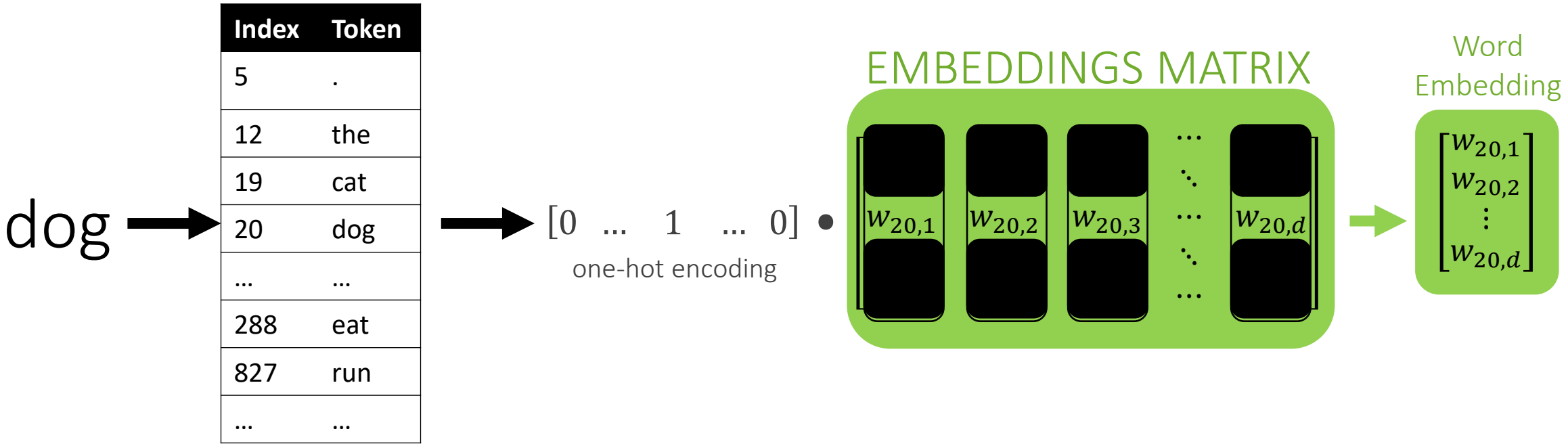
But there are two issues:

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

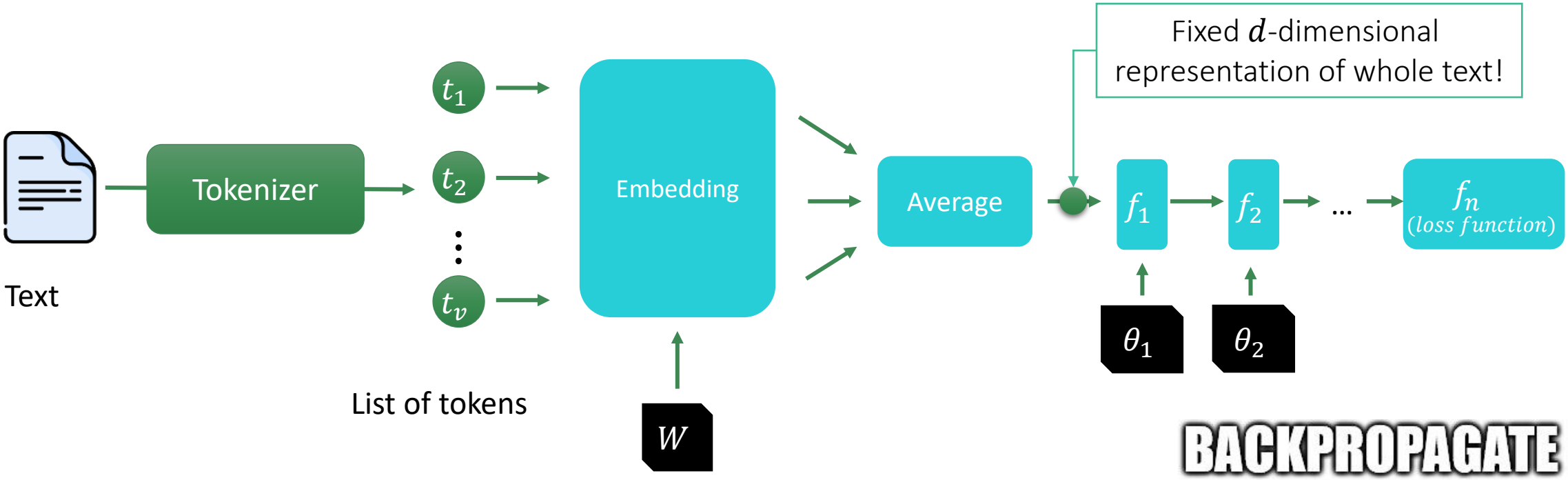
X Each text in a dataset might have different number of tokens

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
  - Select a row from a matrix as a differentiable operation!
  - $d \equiv$ embedding dimension. $[50, 300]$ usually works well.



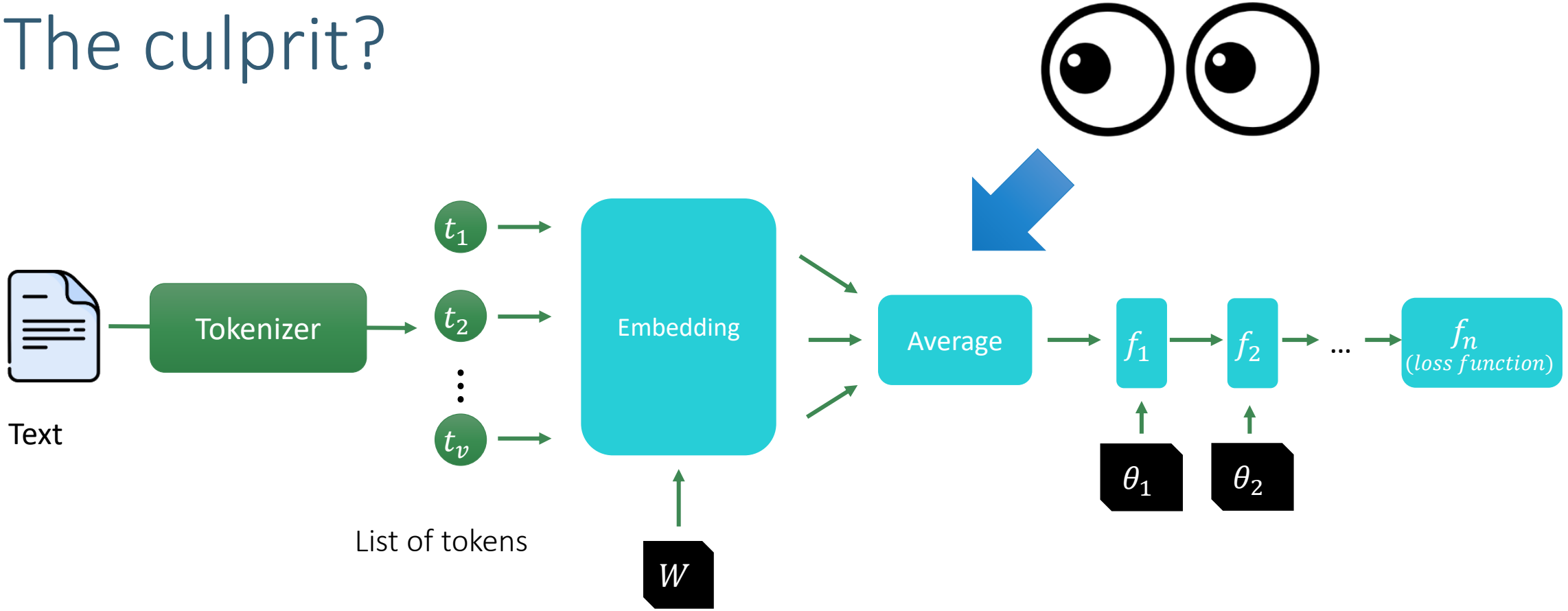
The Continuous Bag of Words model (CBoW)



- 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


EMOSIDO
ENGAÑADO

The culprit?



I had my car cleaned ≠ I had cleaned my car

break the end ≠ end the break

Me río en el baño ≠ Me baño en el río

CBoW performance

✓ Some 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

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

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

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

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

Wikipedia Downloads - https://dumps.wikimedia.org/ Common Crawl - https://commoncrawl.org/

OSCAR (**O**pen **S**uper-large **C**rawled **A**LMAnaCH **C**orpus) - https://oscar-corpus.com/

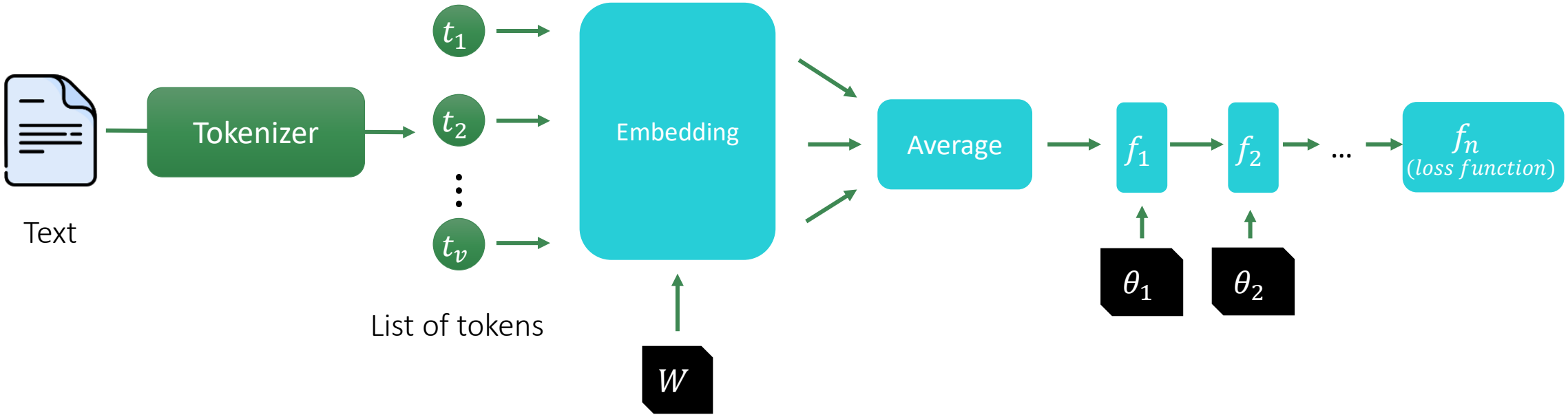
HuggingFace Datasets - https://huggingface.co/datasets

Autoregressive Language Models

Using the definition of conditional probability

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

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

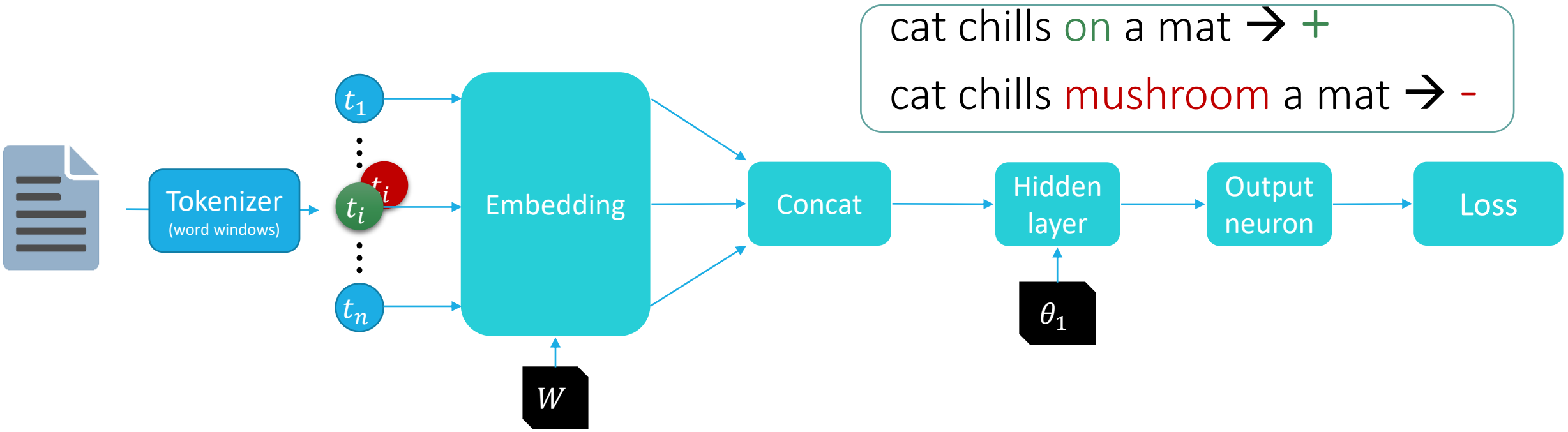


Language model example: word2vec

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
carnivorous 234
cetaceans 234
cormorant 234
coyotes 234
crocodile 234
crocodiles 234
crustaceans 234
cultivated 234
danios 234
```

```
acceptance 412
argue 412
argues 412
arguing 412
argument 412
arguments 412
belief 412
believe 412
challenge 412
claim 412
```

word2vec: semantic algebra

King − man + woman = queen

Obama − USA + Russia = Putin

paella − Spain + Italy = risotto

Cristiano − Madrid + Barcelona = Messi

fastText

Download pre-trained models

English word vectors

Pre-trained on English webcrawl and Wikipedia

Multi-lingual word vectors

Pre-trained models for 157 different languages

https://fasttext.cc/

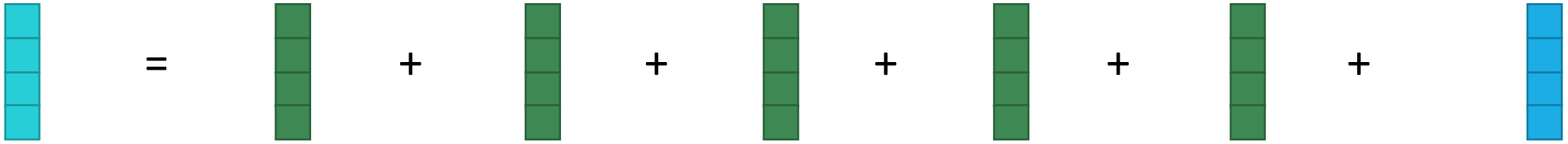
Fasttext embeddings

✓ 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>

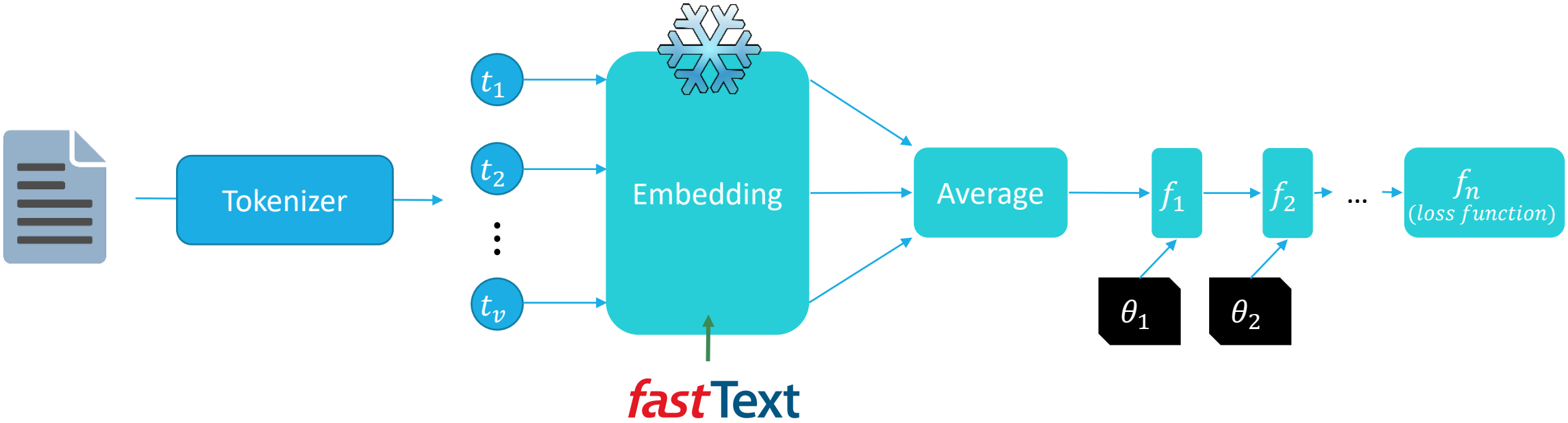


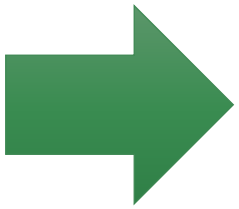
Efficient implementations allow learning a more precise language model with extremely large corpora: Common Crawl.

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





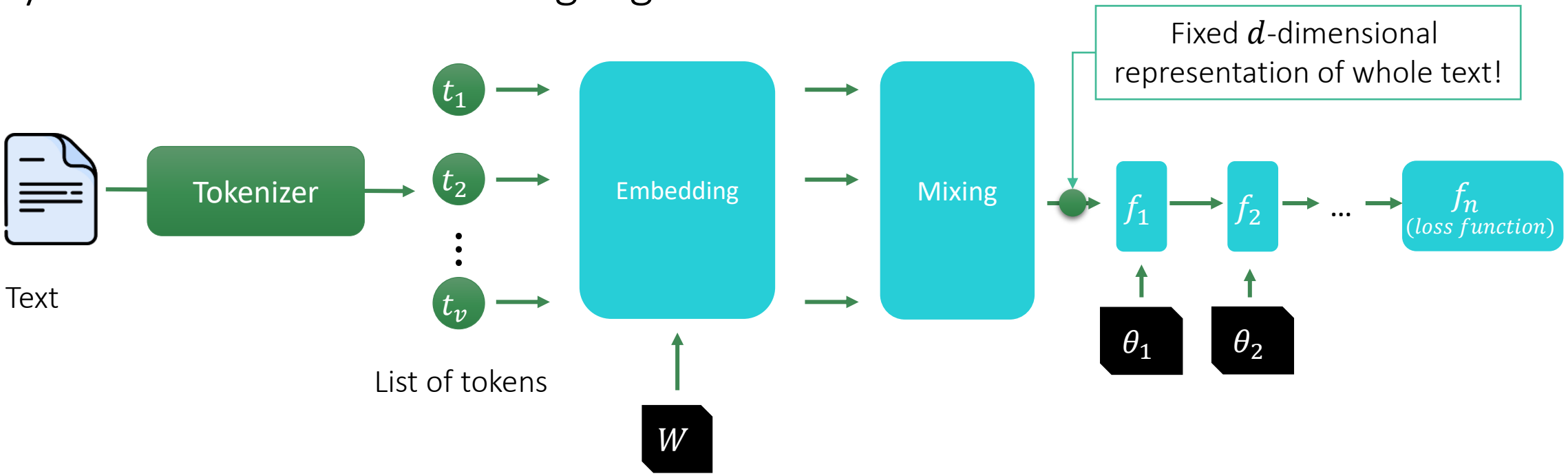
word2vec: https://code.google.com/p/word2vec/

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

FastText: https://fasttext.cc/

Better token combinations: mixing models

We will use a specialized layer (or group of layers) to mix the embeddings in a way that makes sense for language.



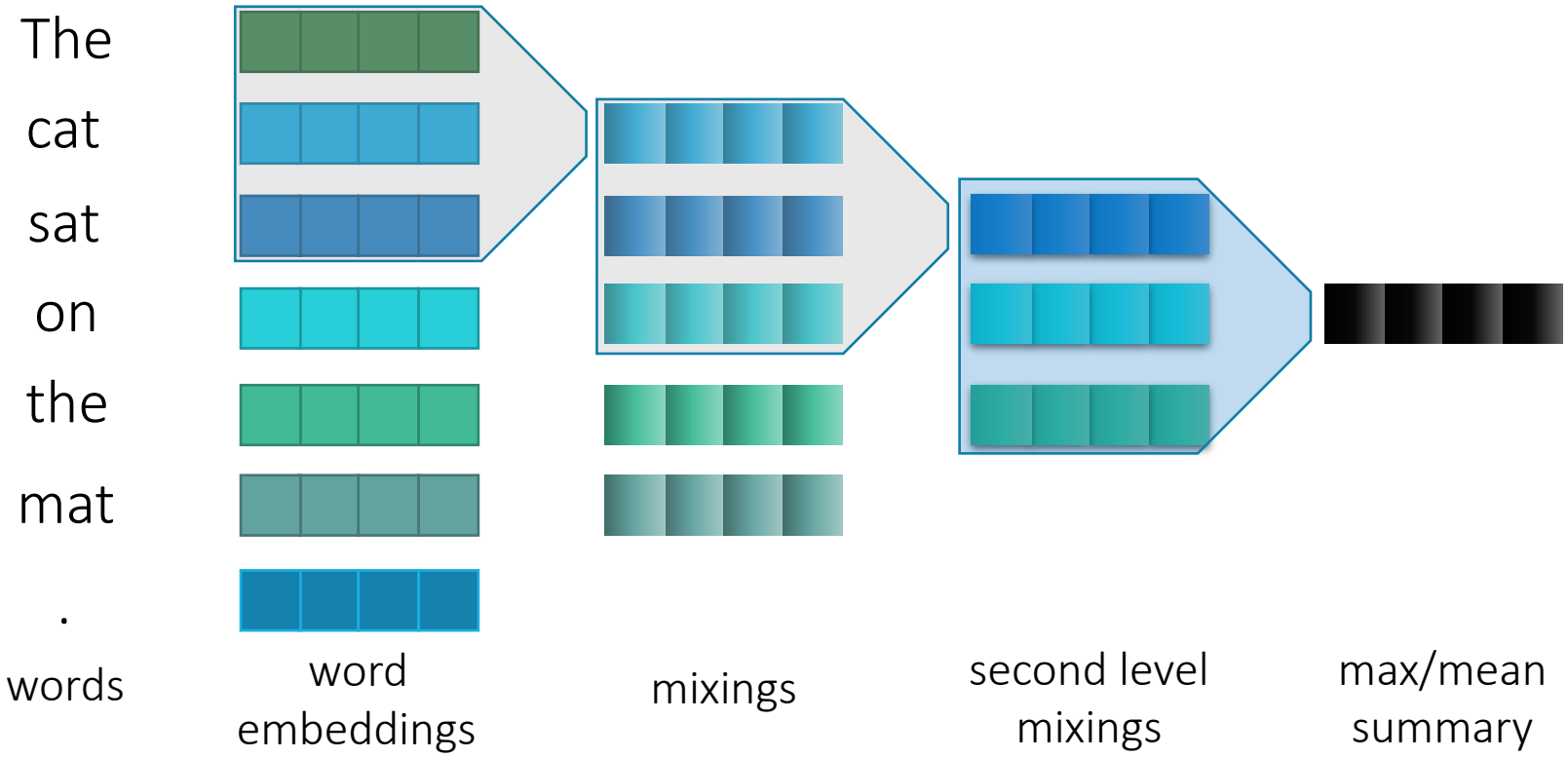
- Convolutions
- Recurrent layers
- Transformers

Convolutions over embeddings

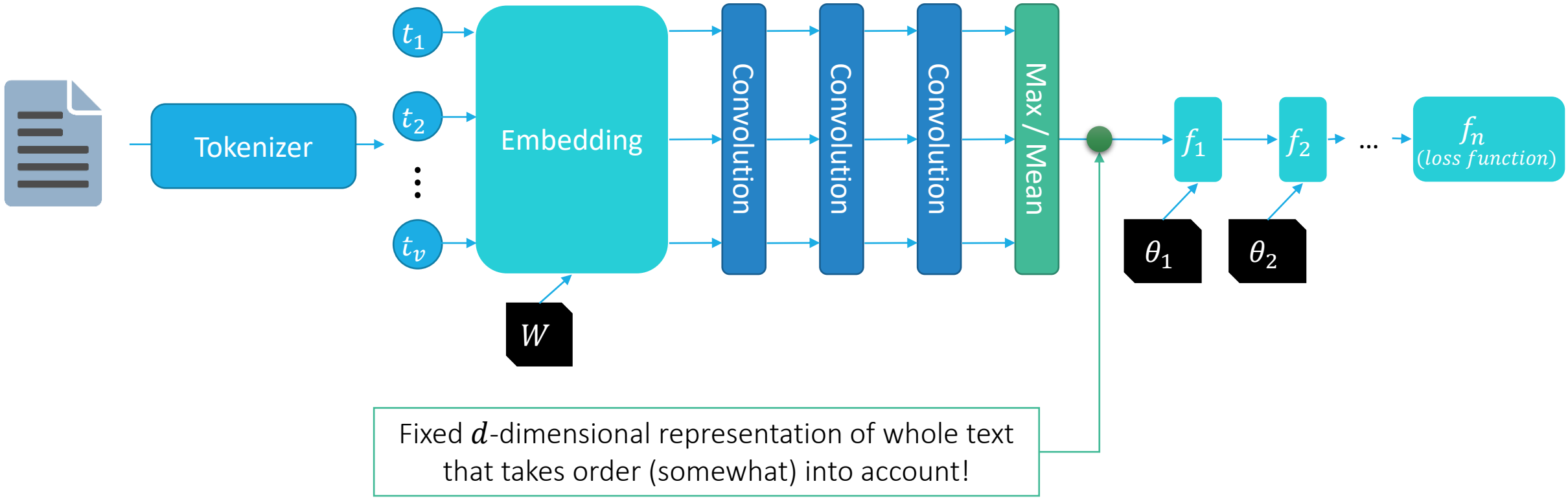
Convolutions are helpful to mix the representation of a word with the representation of neighboring words.

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.



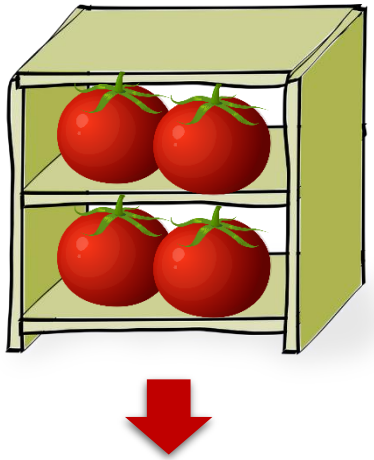
Stacked convolutions: network diagram

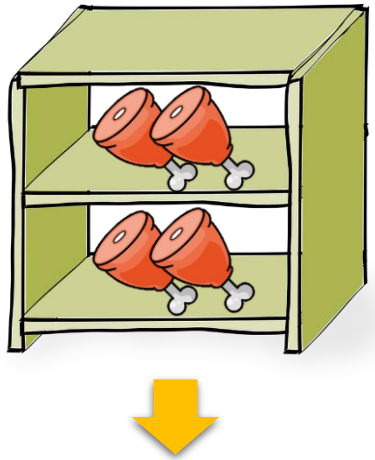


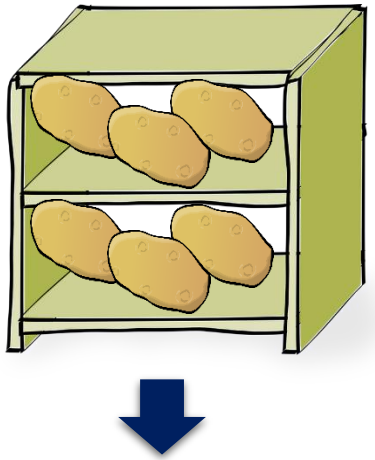
Better combinations: let's cook a stew!

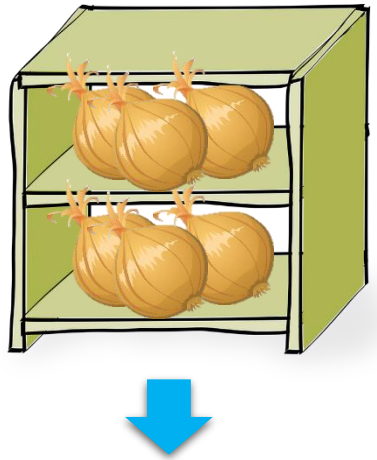
Inputs:

(Nothing)

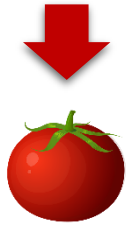


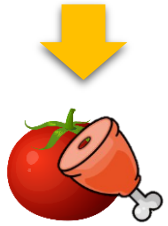


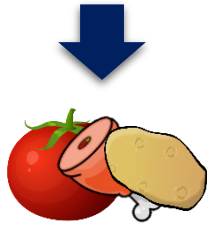


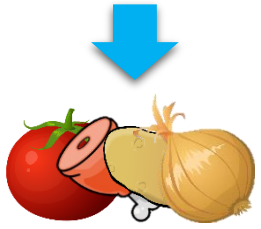


Outputs:

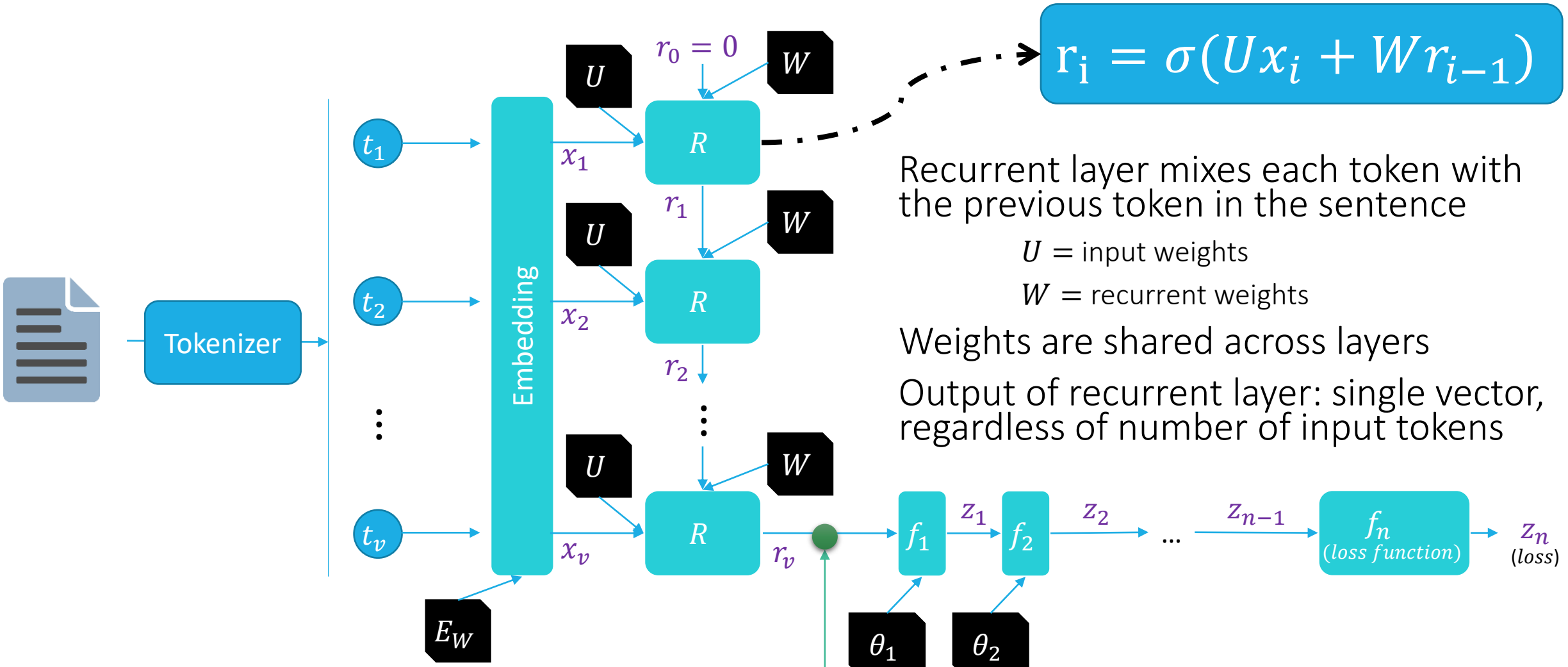




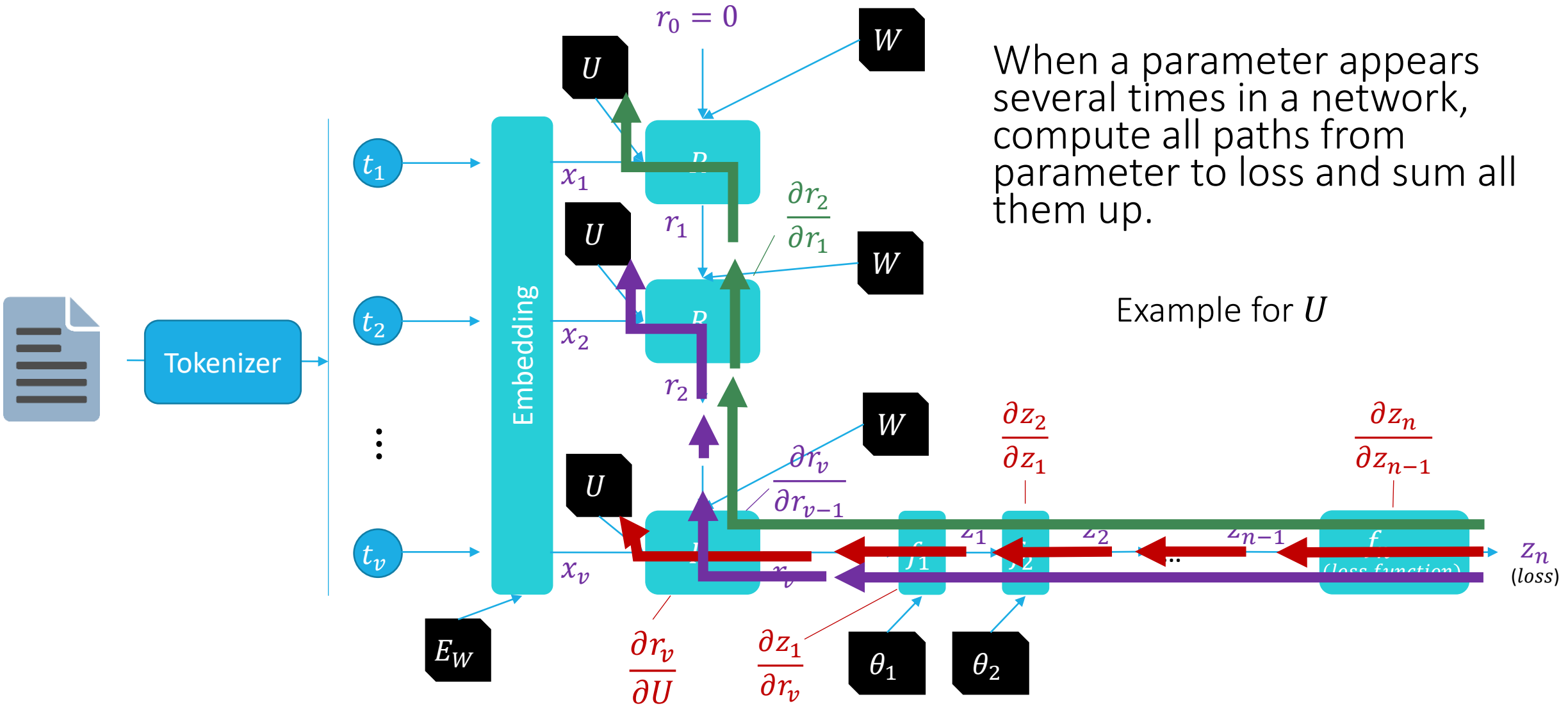




Better token combinations: recurrent layers



Backpropagation with recurrent layers



When a parameter appears several times in a network, compute all paths from parameter to loss and sum all them up.

Example for U

$$\frac{\partial F}{\partial U} = \frac{\partial z_n}{\partial z_{n-1}} \cdots \frac{\partial z_2}{\partial z_1} \frac{\partial z_1}{\partial r_v} \frac{\partial r_v}{\partial U} + \frac{\partial z_n}{\partial z_{n-1}} \cdots \frac{\partial z_2}{\partial z_1} \frac{\partial z_1}{\partial r_v} \frac{\partial r_v}{\partial r_{v-1}} \cdots \frac{\partial r_2}{\partial U} + \frac{\partial z_n}{\partial z_{n-1}} \cdots \frac{\partial z_2}{\partial z_1} \frac{\partial z_1}{\partial r_v} \frac{\partial r_v}{\partial r_{v-1}} \cdots \frac{\partial r_2}{\partial r_1} \frac{\partial r_1}{\partial U} + \cdots$$

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!

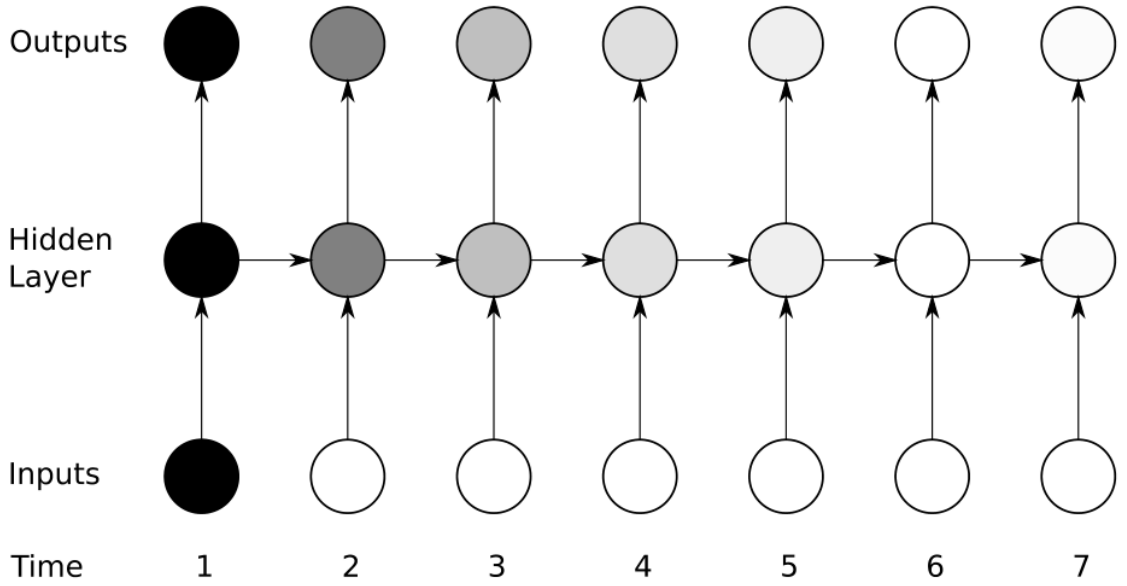


Figure 4.1: **The vanishing gradient problem for RNNs.** The shading of the nodes in the unfolded network indicates their sensitivity to the inputs at time one (the darker the shade, the greater the sensitivity). The sensitivity decays over time as new inputs overwrite the activations of the hidden layer,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

Graves - Supervised Sequence Labelling with Recurrent Neural Networks

Dealing with vanishing gradients

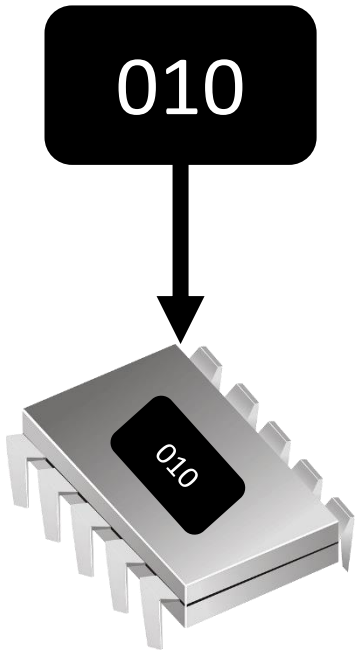
- To deal with vanishing gradients, many approaches have been proposed
  - Smarter weights initialization
  - Rectified Linear Units
  - Non-gradient based methods (e.g. discrete error propagation)

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

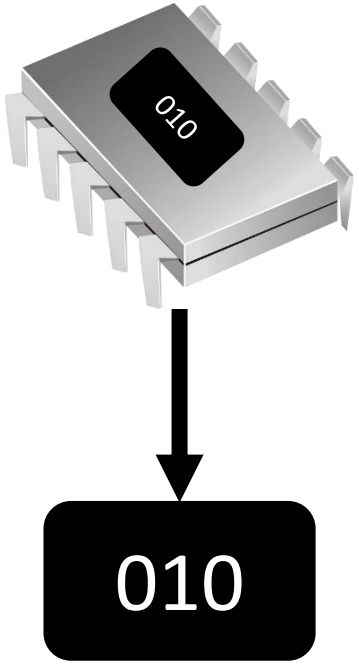
About memories

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

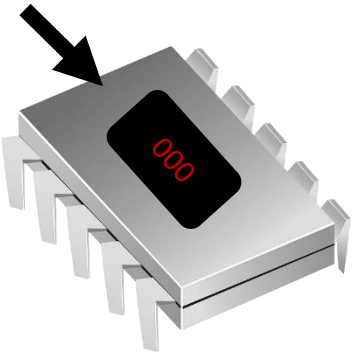
Write



Read



Reset

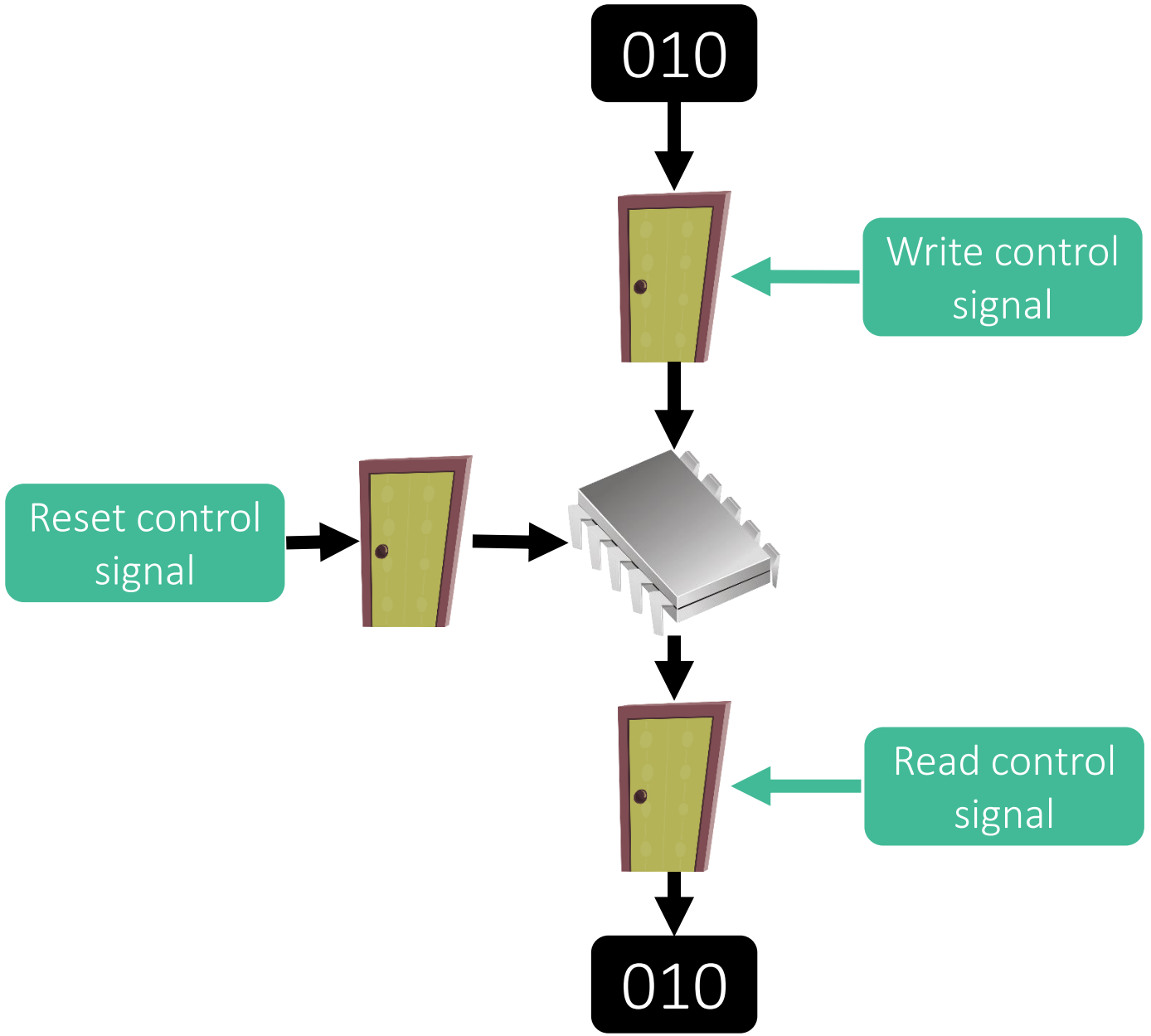


Gated memories

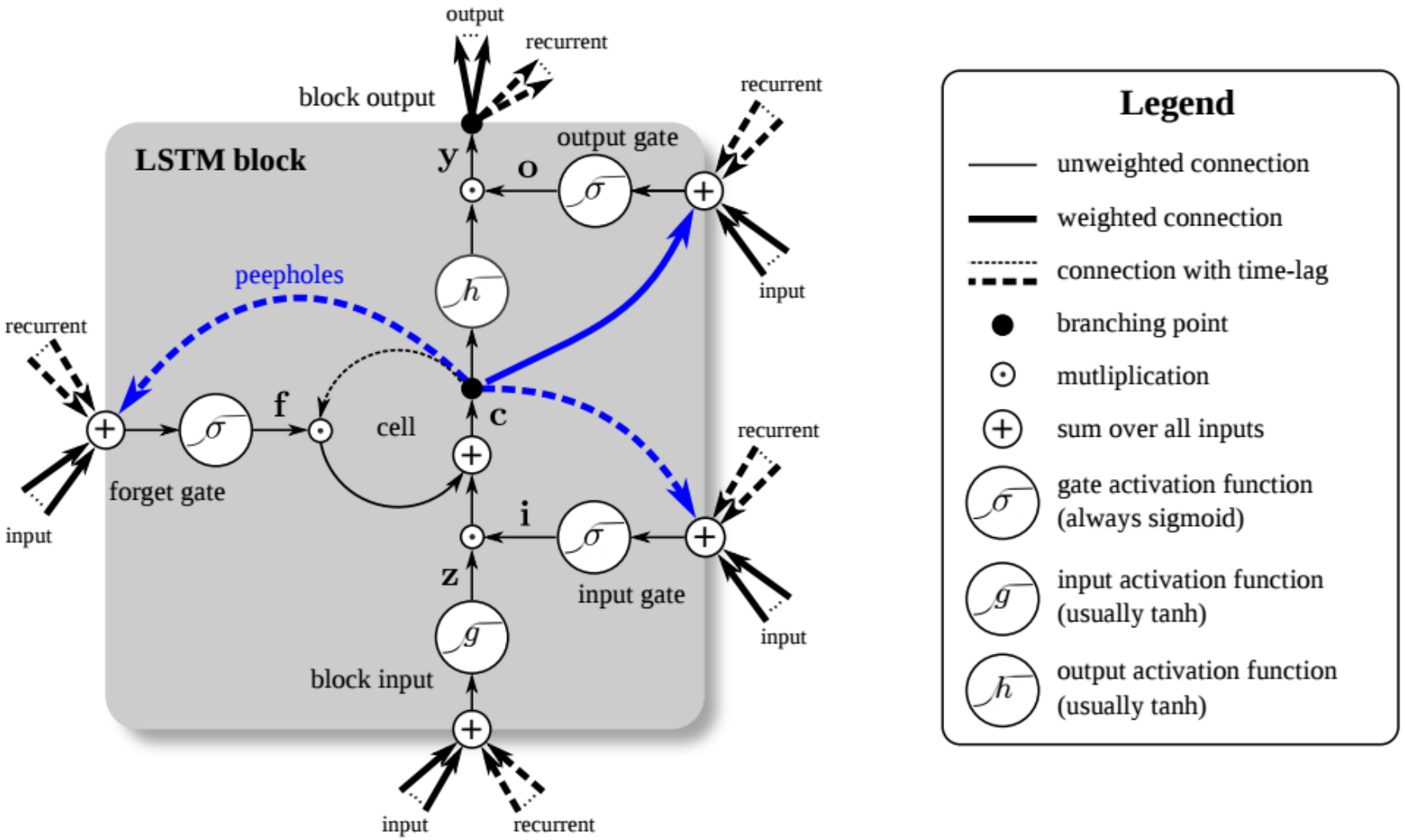
The three operations can be combined, together with some gates that open/close the execution of each operation.

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!

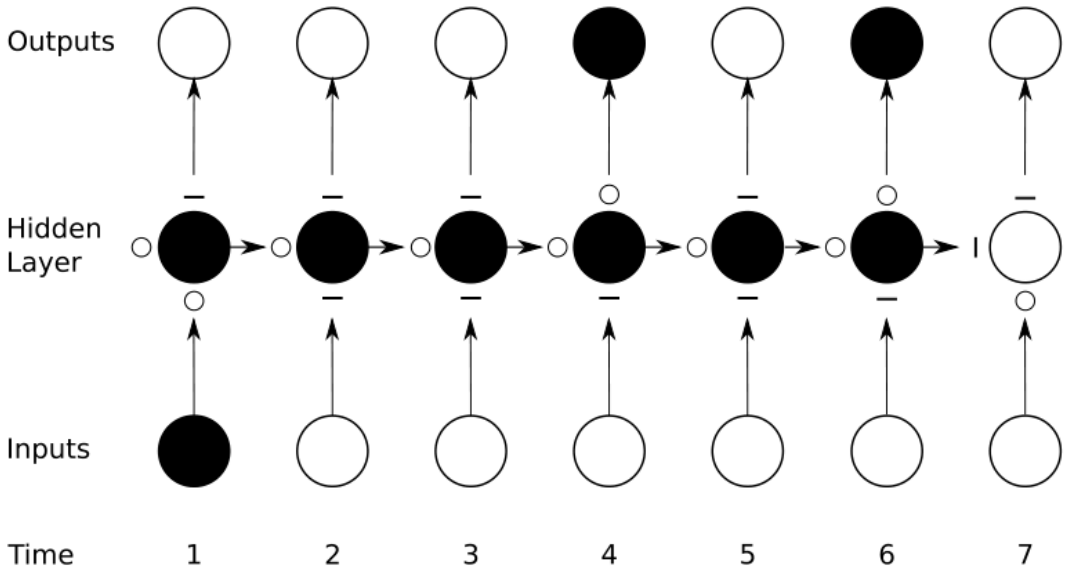
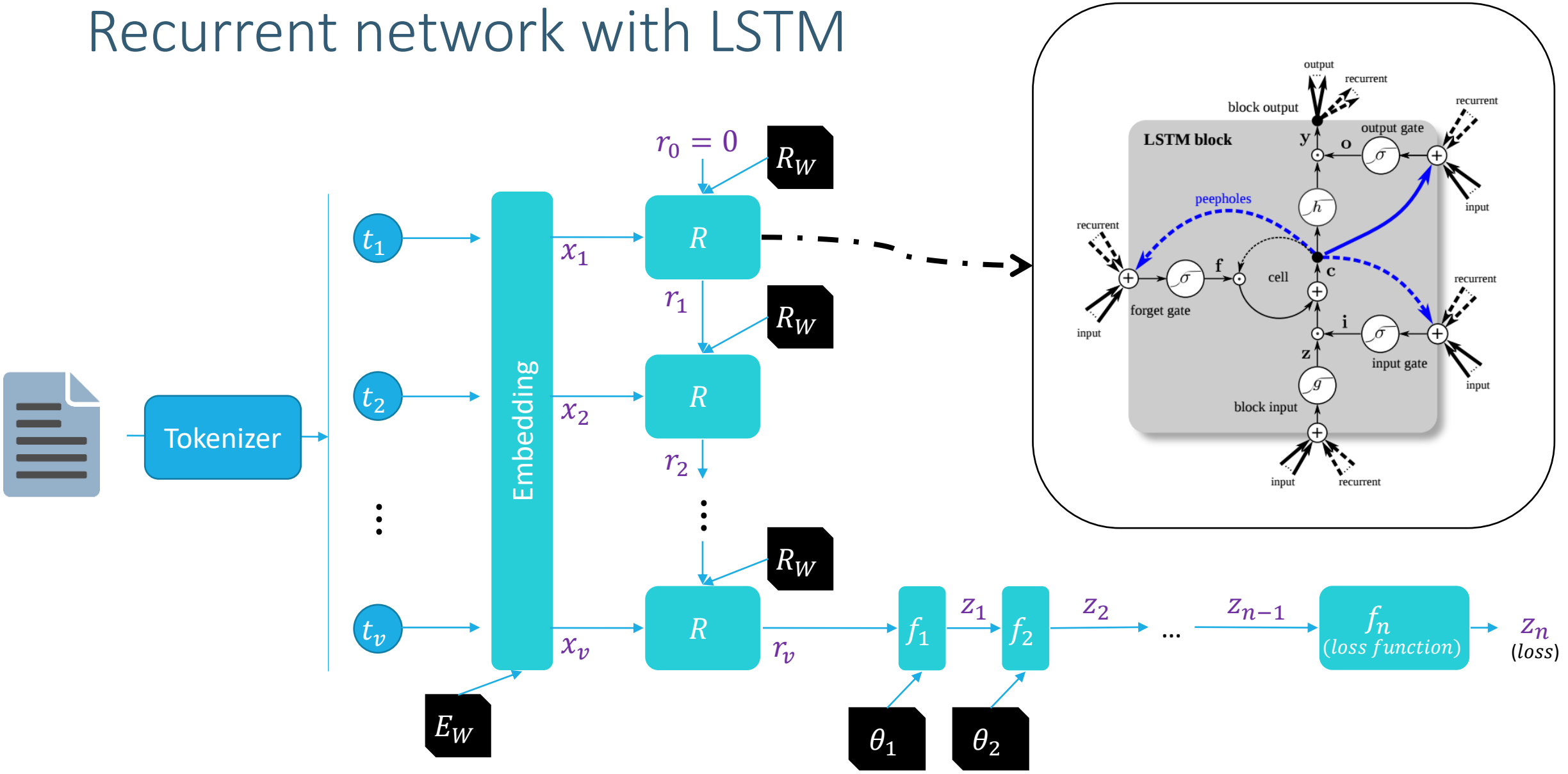


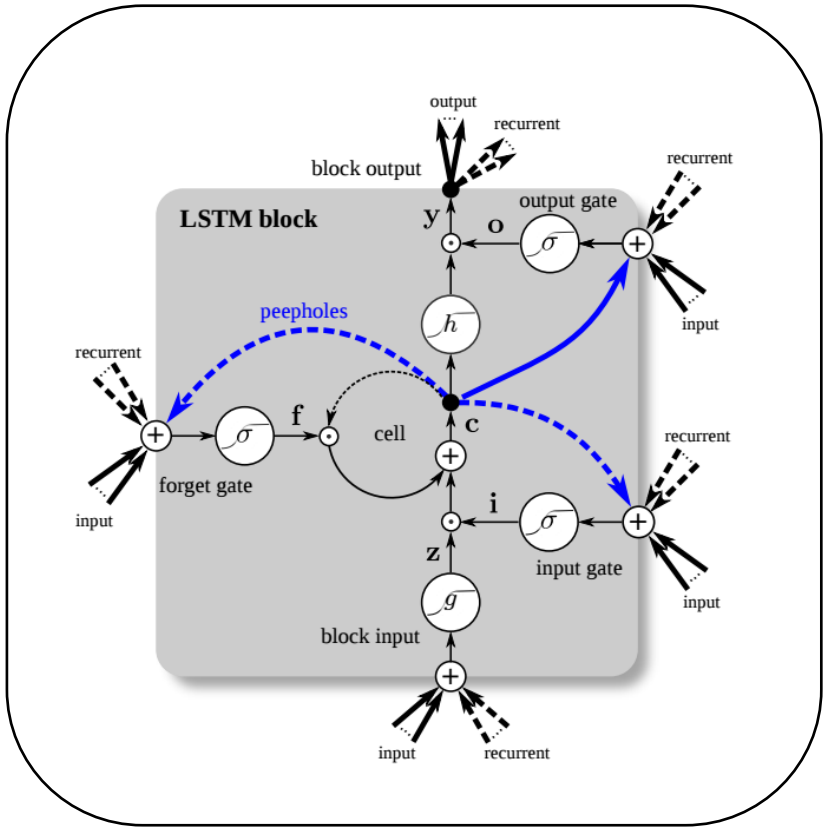
Figure 4.4: **Preservation of gradient information by LSTM.** As in Figure 4.1 the shading of the nodes indicates their sensitivity to the inputs at time one; in this case the black nodes are maximally sensitive and the white nodes are entirely insensitive. The state of the input, forget, and output 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

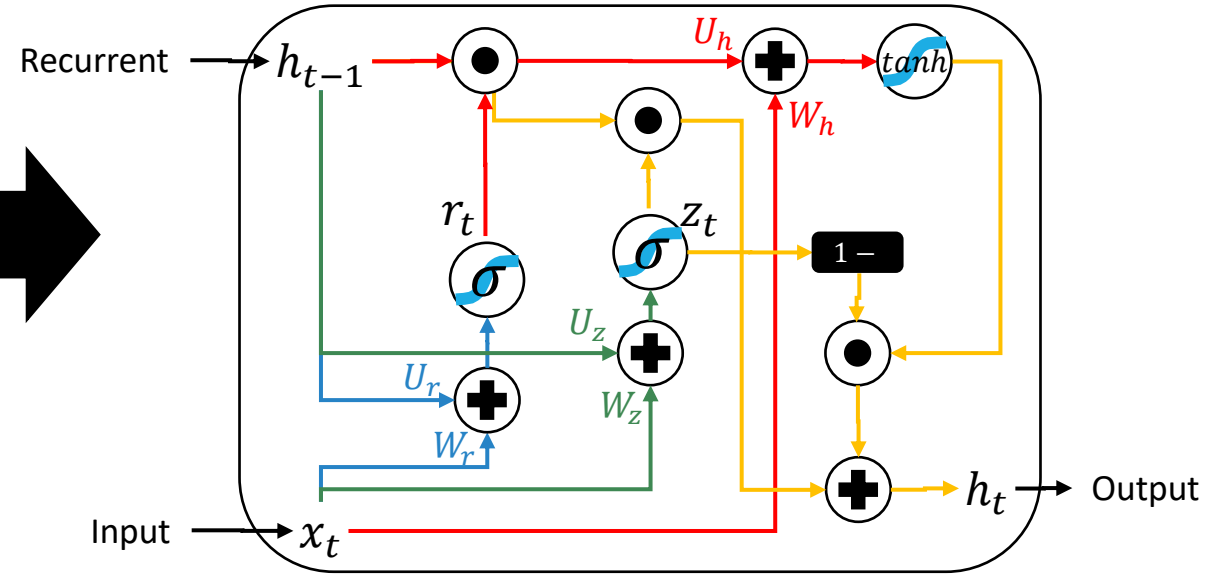
Recurrent network with LSTM



Gated Recurrent Units (GRU)



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

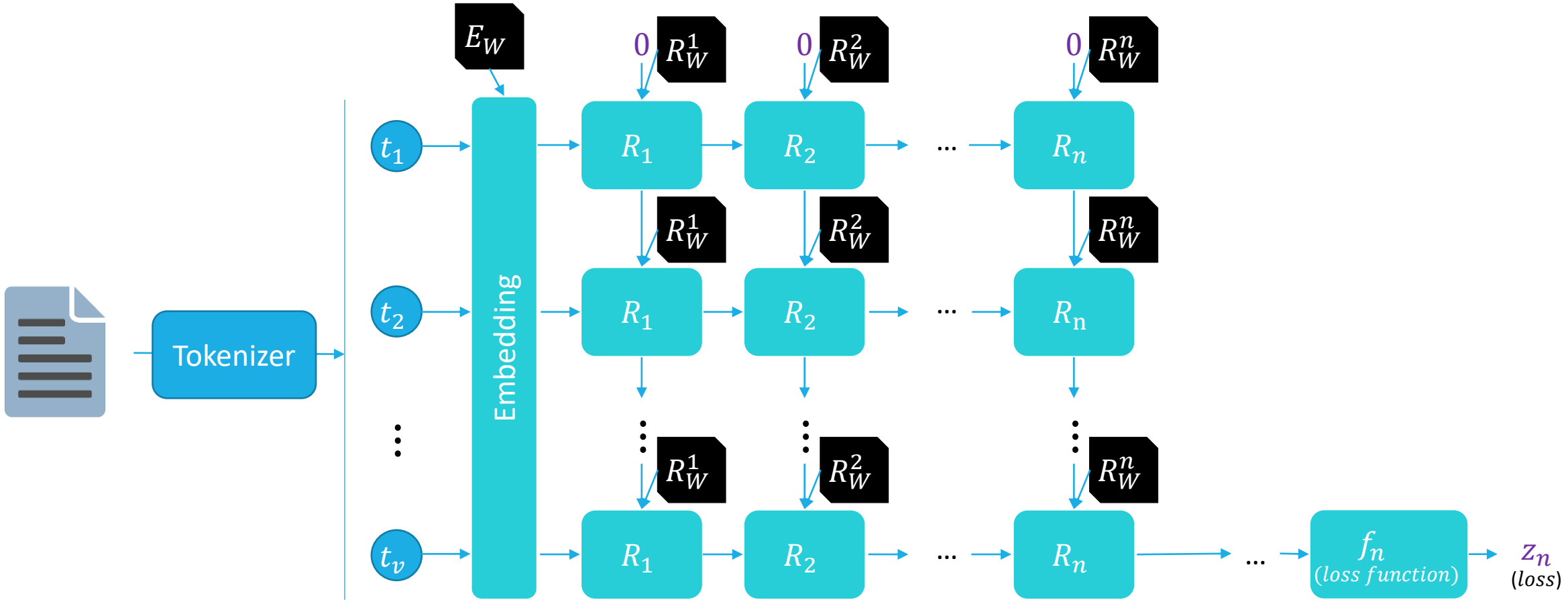


- Simplified version of the LSTM cell
  - Combine "write" and "reset" gates into a single "update" gate. No "read" gate: output values h_t are the memory values themselves
- Lower computational cost
- Works better than LSTM in smaller datasets

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

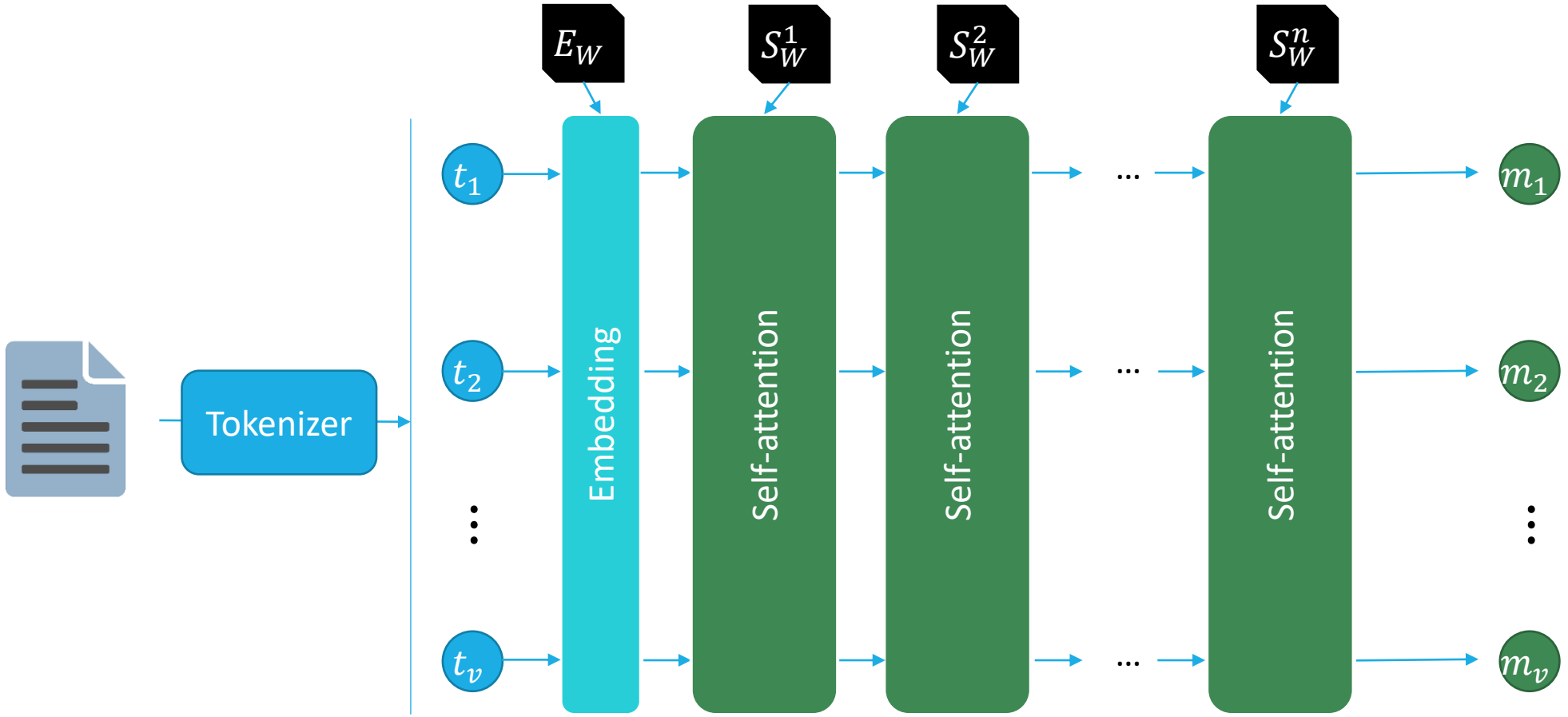
Colah – Understanding LSTM Networks - http://colah.github.io/posts/2015-08-Understanding-LSTMs/

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

Self-attention



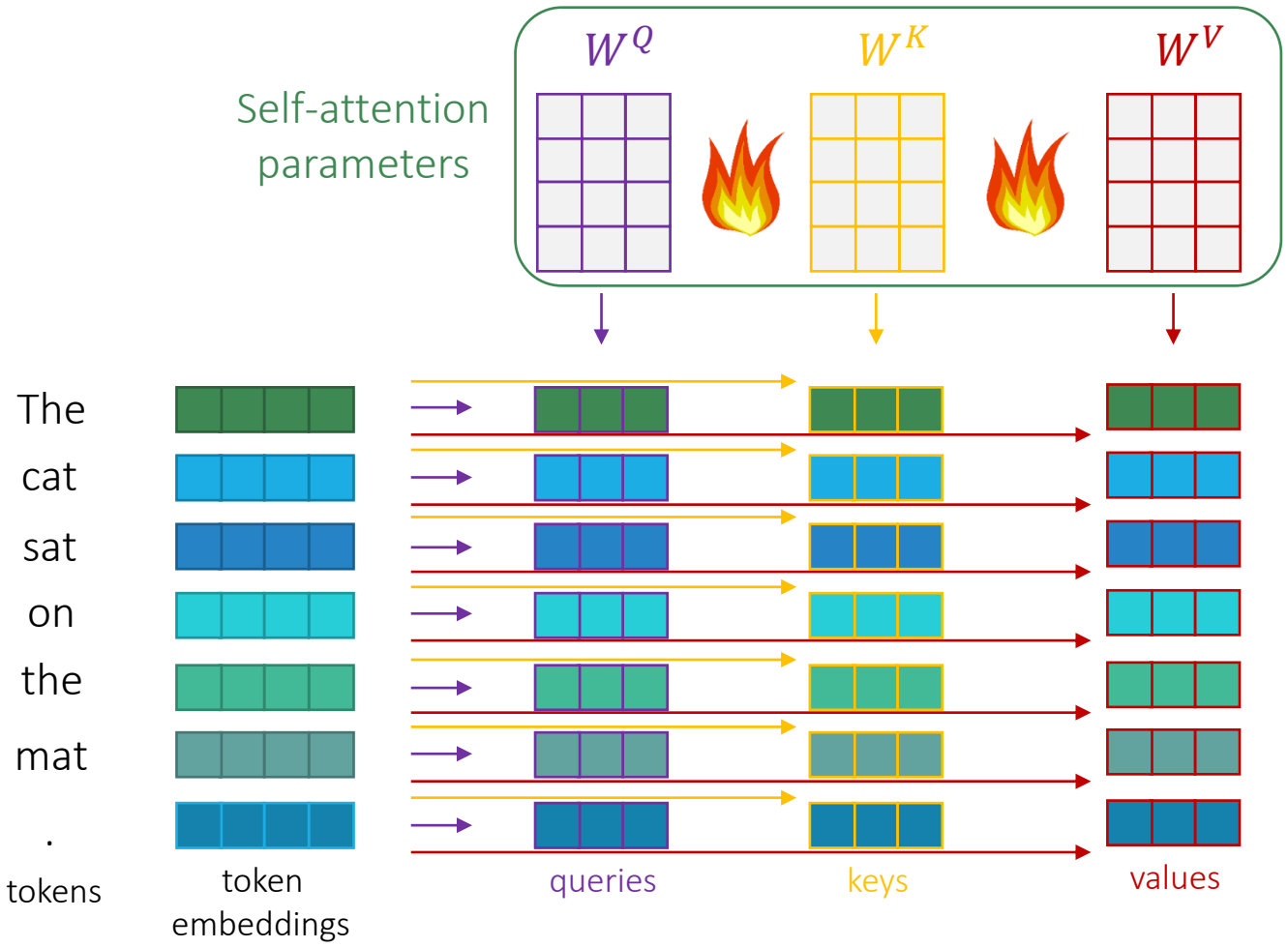
- Improving over recurrent networks, the self-attention layer mixes the representation of every token in the document with every other token, and produces a new embedding for each token.
- Contextualized embeddings.

Self-attention is like…

Tinder

… but for words

Self-attention in detail

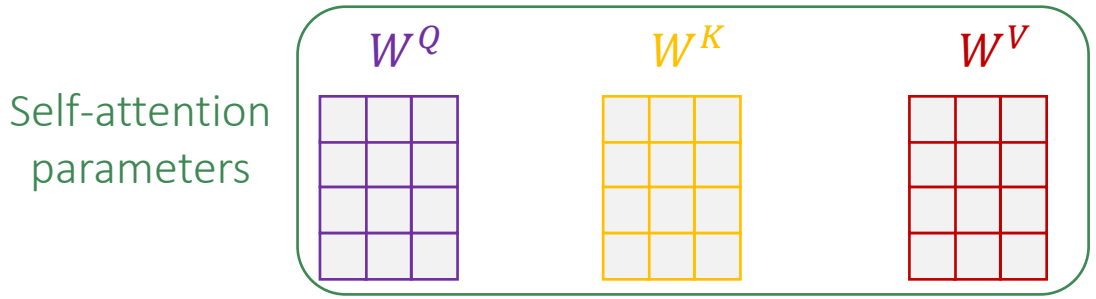


- Multiply each token embedding by matrices W^Q, W^K and W^V to produce a query, key and value vector for each token.
- Query vectors: what this token is "looking for" in other tokens
- Key vectors: what this token "can offer" to other tokens
- Relevance score: how aligned is a query vector for a token 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.

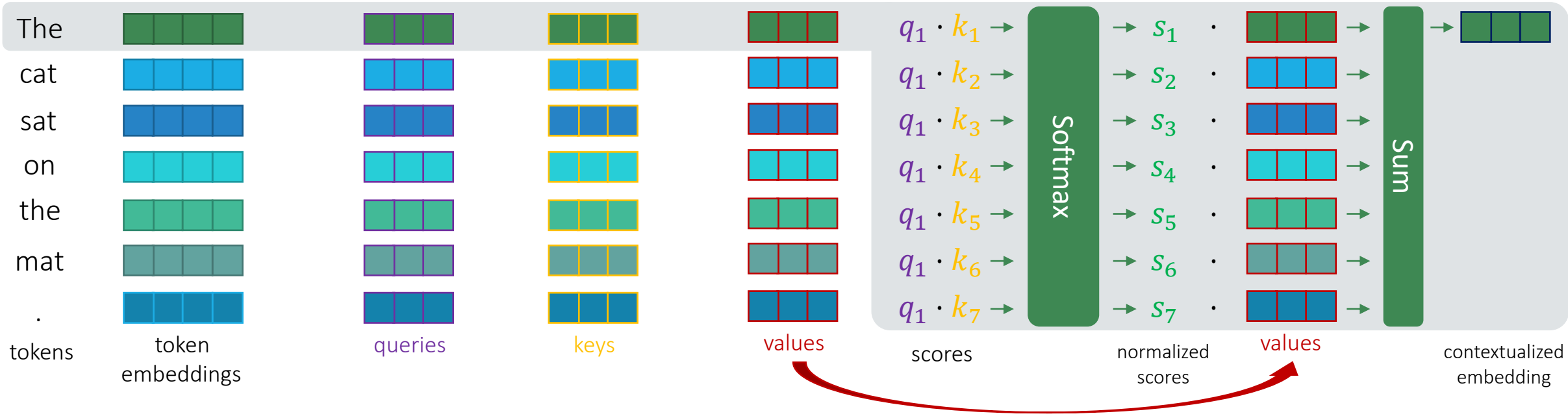
Jay Allamar – The Illustrated Transformer - https://jalammar.github.io/illustrated-transformer/

Vaswani et al – Attention is All You need

Self-attention scores for a token



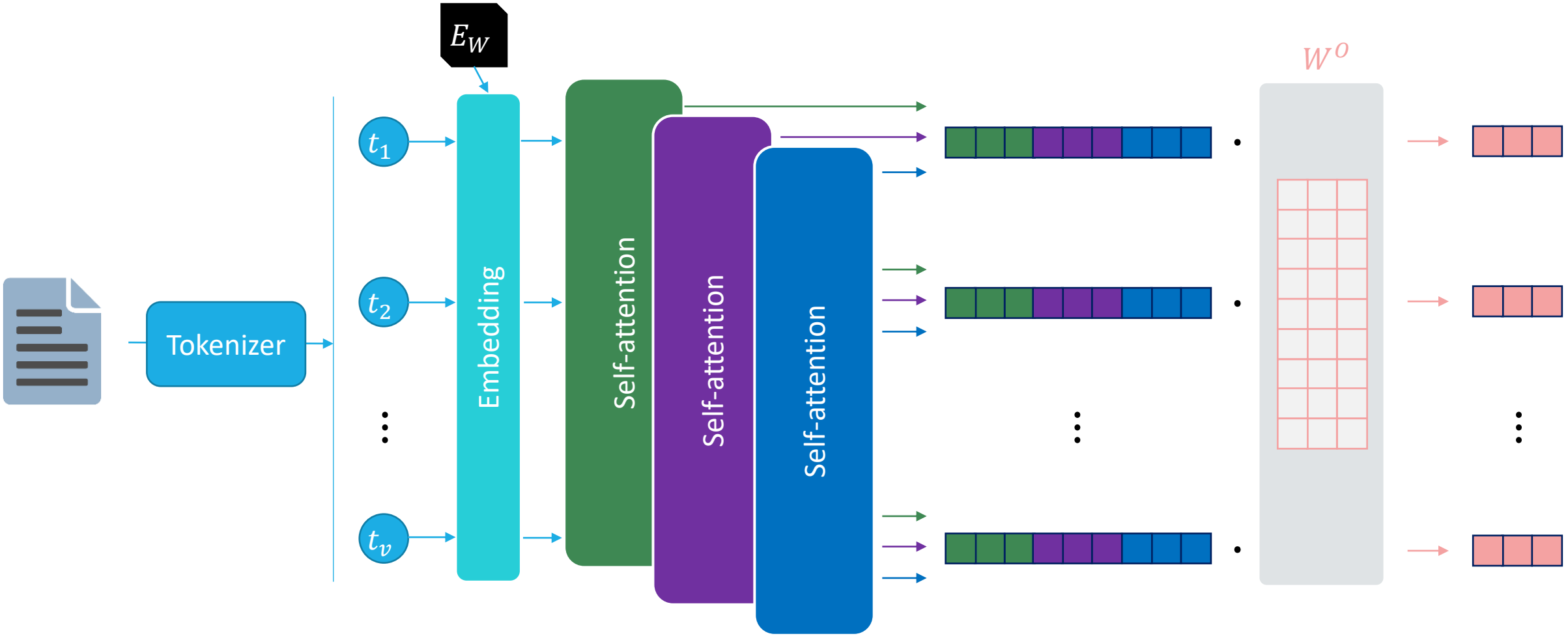
- The contextualized embedding for each token mixes value vectors from all words, weighted by query · key scores.
- Flow for first token:



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

Vaswani et al – Attention is All You need

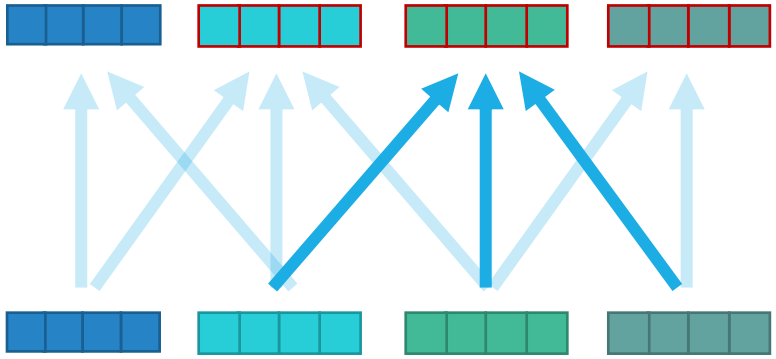
Multi-headed self-attention



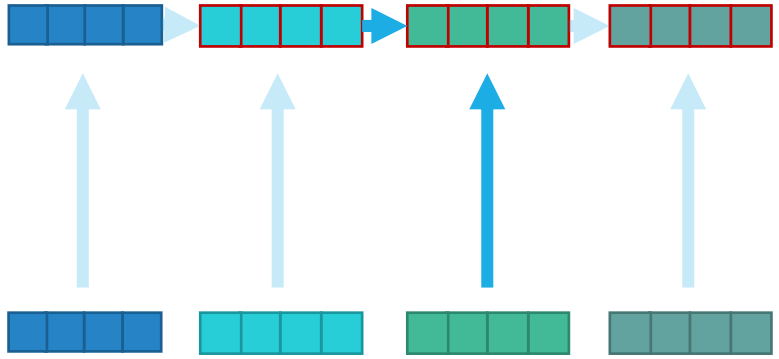
- Multiple self-attention blocks can be used in parallel, similarly to multiple convolutional kernels.
- For n blocks (heads) n contextualized embeddings are obtained for each token.
- Concatenate embeddings and mix using an output matrix W^O to recover a single embedding per token.

Summary of mixing models

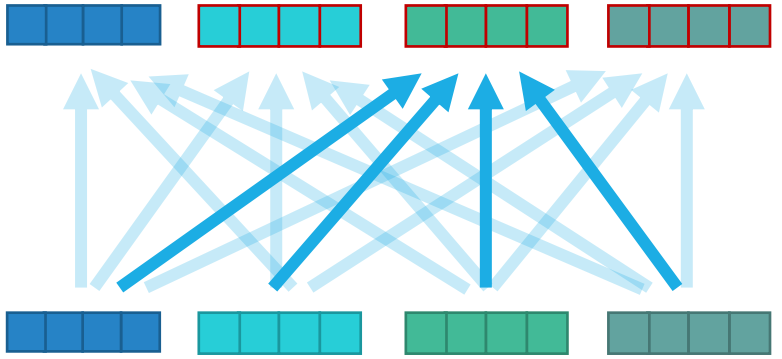
Mixing model	Operation	Sensitive to tokens relative ordering?	Sensitive to tokens absolute position in the document?	Mixing operation adapts to input tokens?
Convolutional	Mix each token with neighbouring tokens	✓	✗	✗
Recurrent	Mix each token with new representation of previous token	✓	✓	✓
Self-attention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



Convolutional



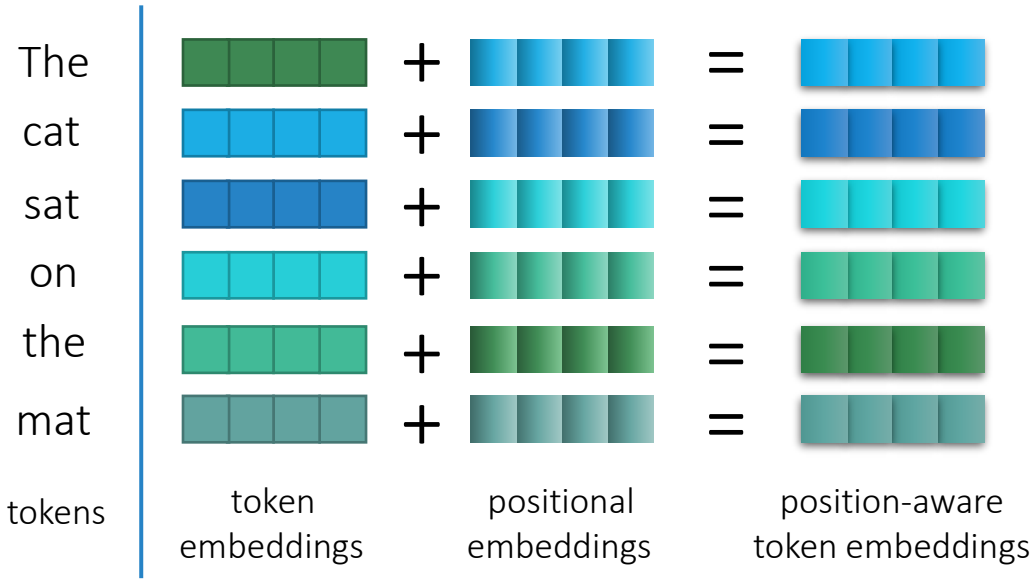
Recurrent

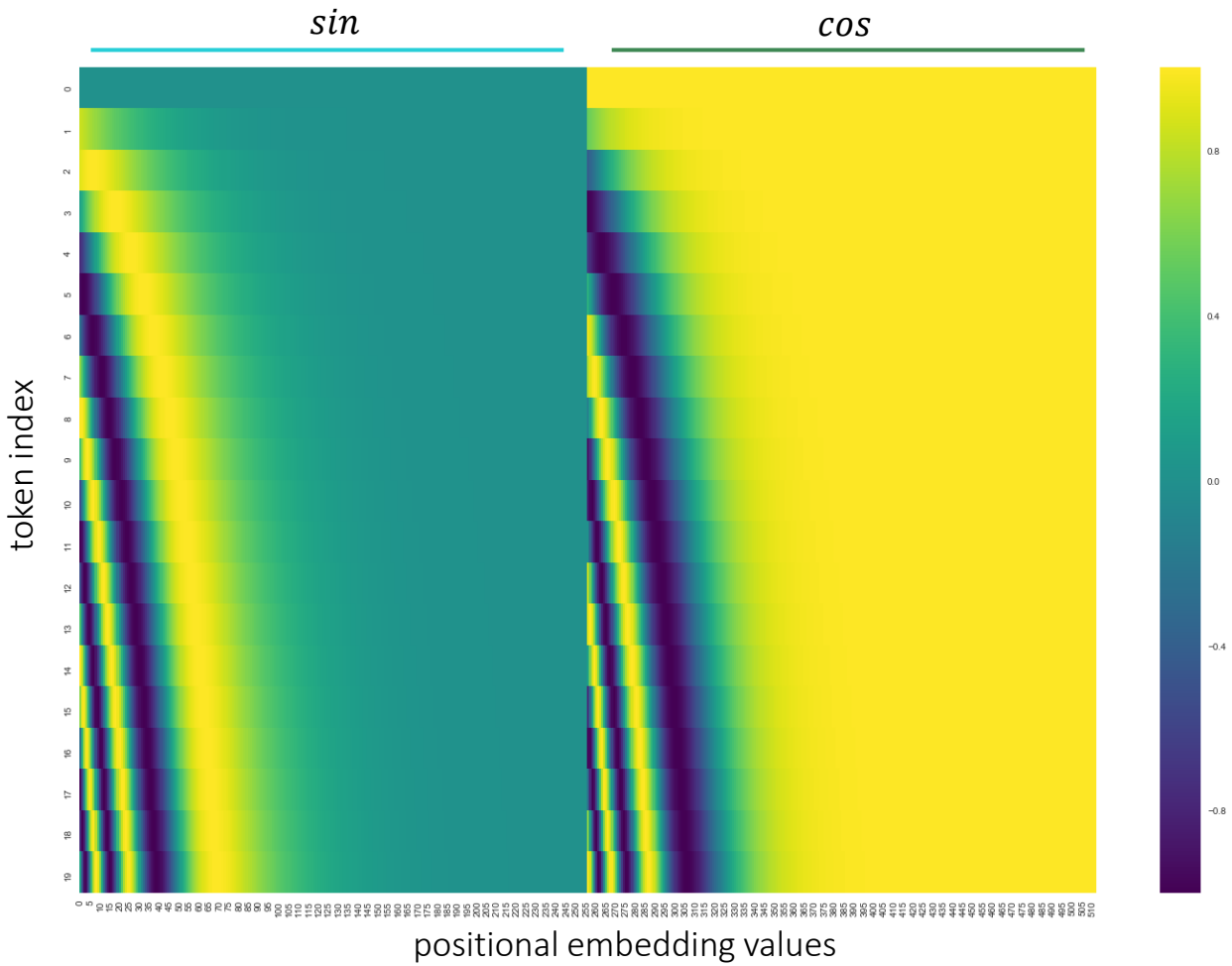


Self-attention

Positional embeddings

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





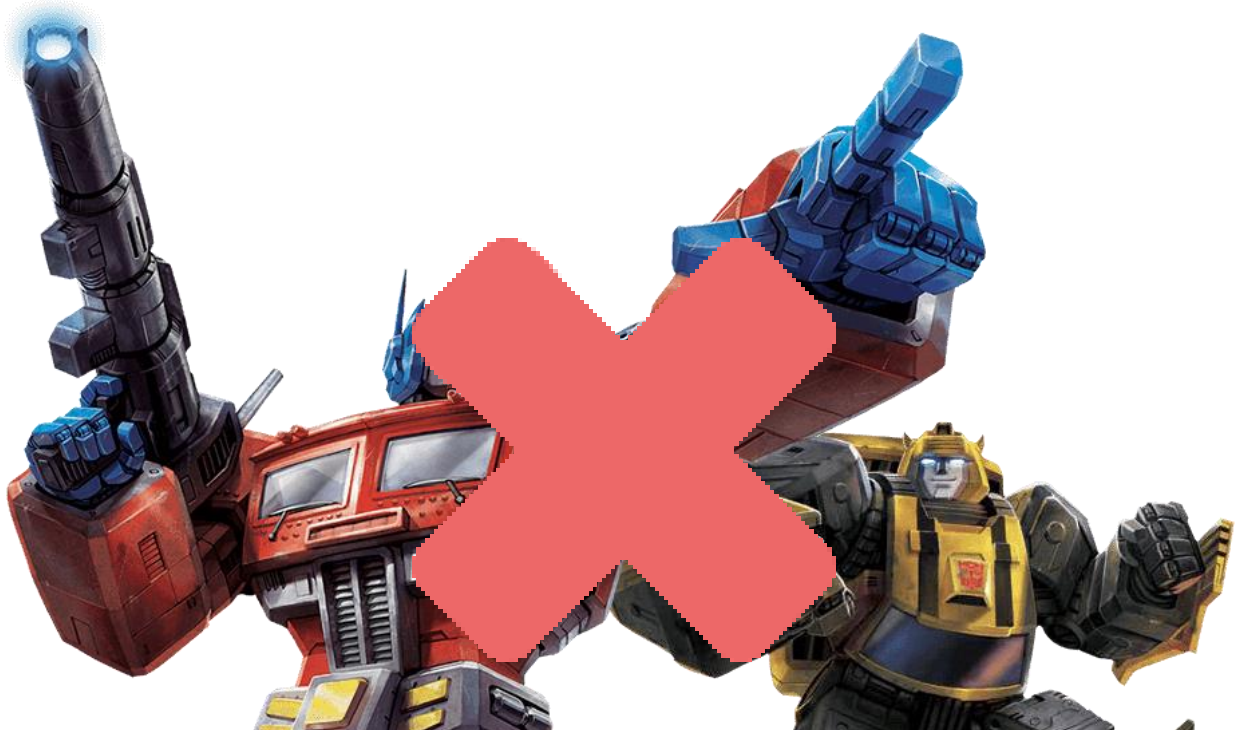
Positional embeddings values are computed as sin and cos of the position of the word in the document. Each embedding dimension i accounts for a sinusoid of a different frequency, normalized by maximum document length. Positional embeddings have the same size d_{model} has token embeddings.

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

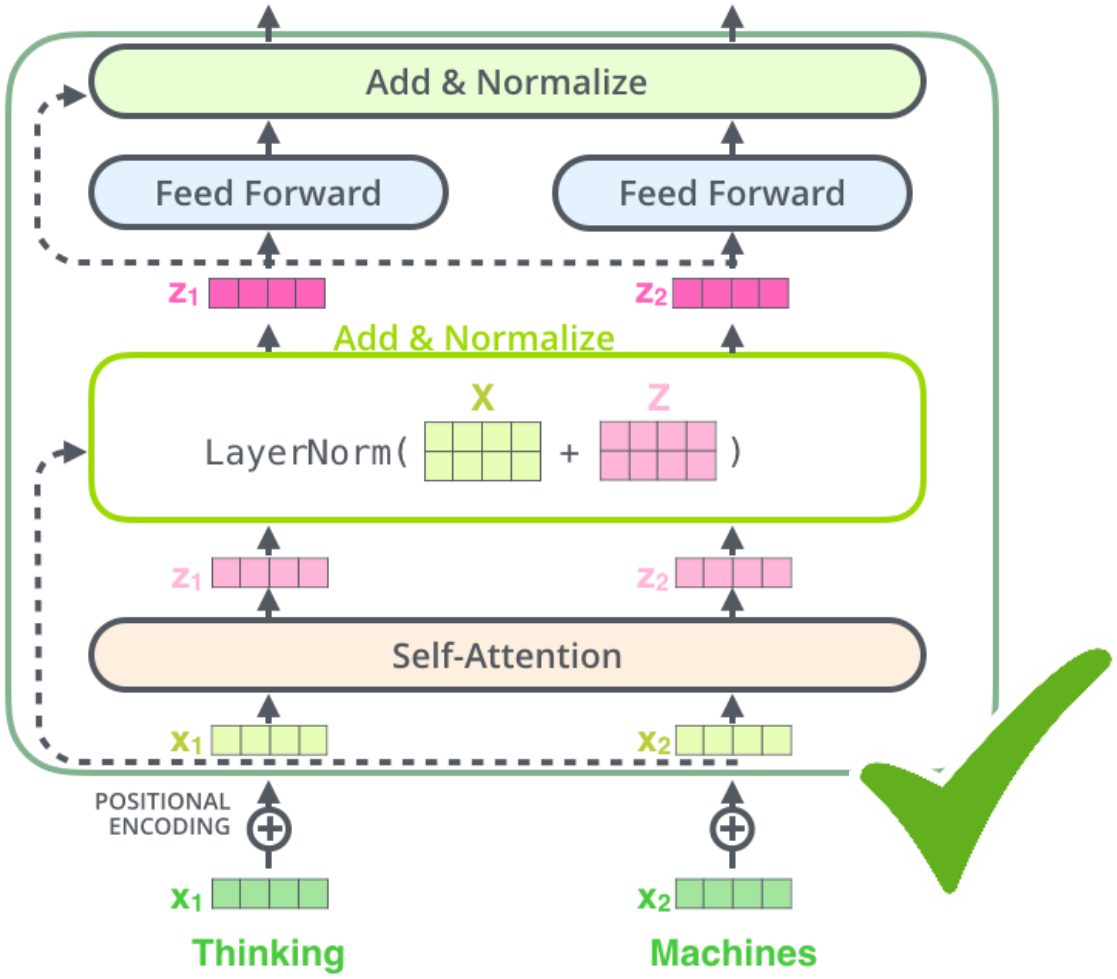
Jay Allamar – The Illustrated Transformer - https://jalammar.github.io/illustrated-transformer/

Vaswani et al – Attention is All You need

This new breed of neural networks shall be called…



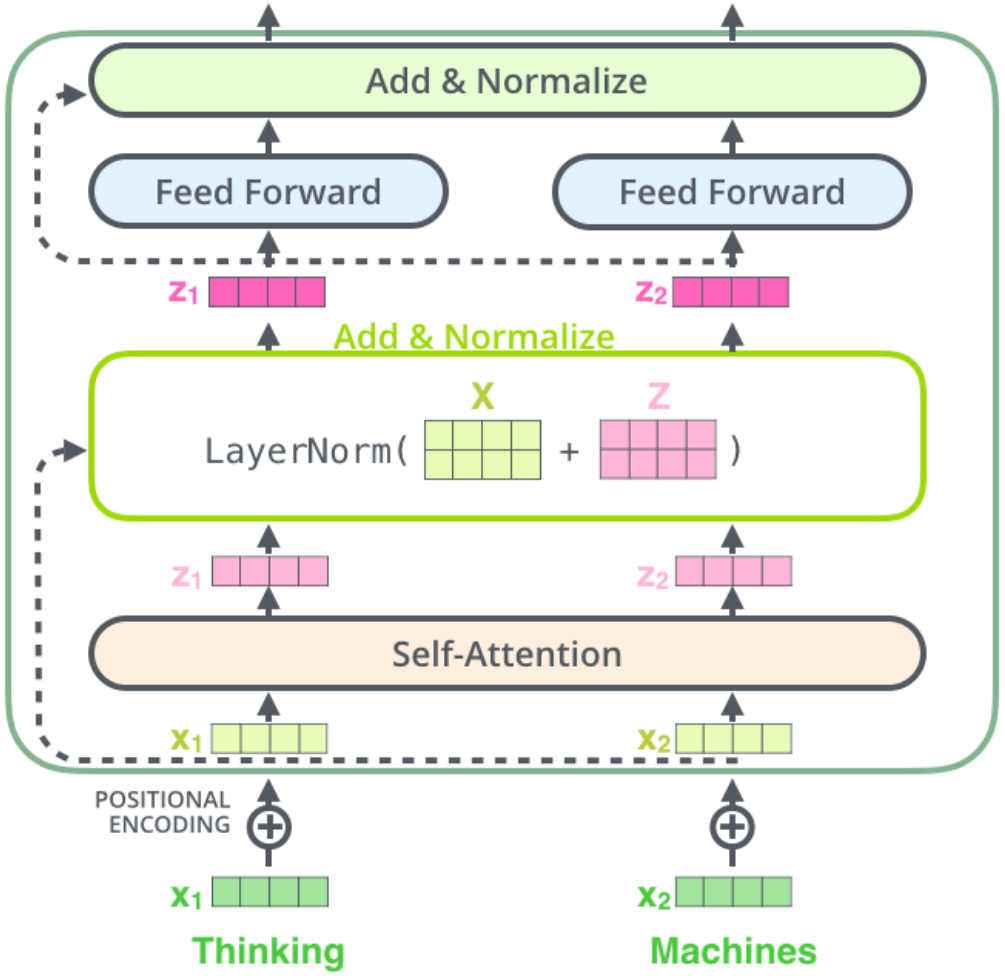
TRANSFORMERS



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

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

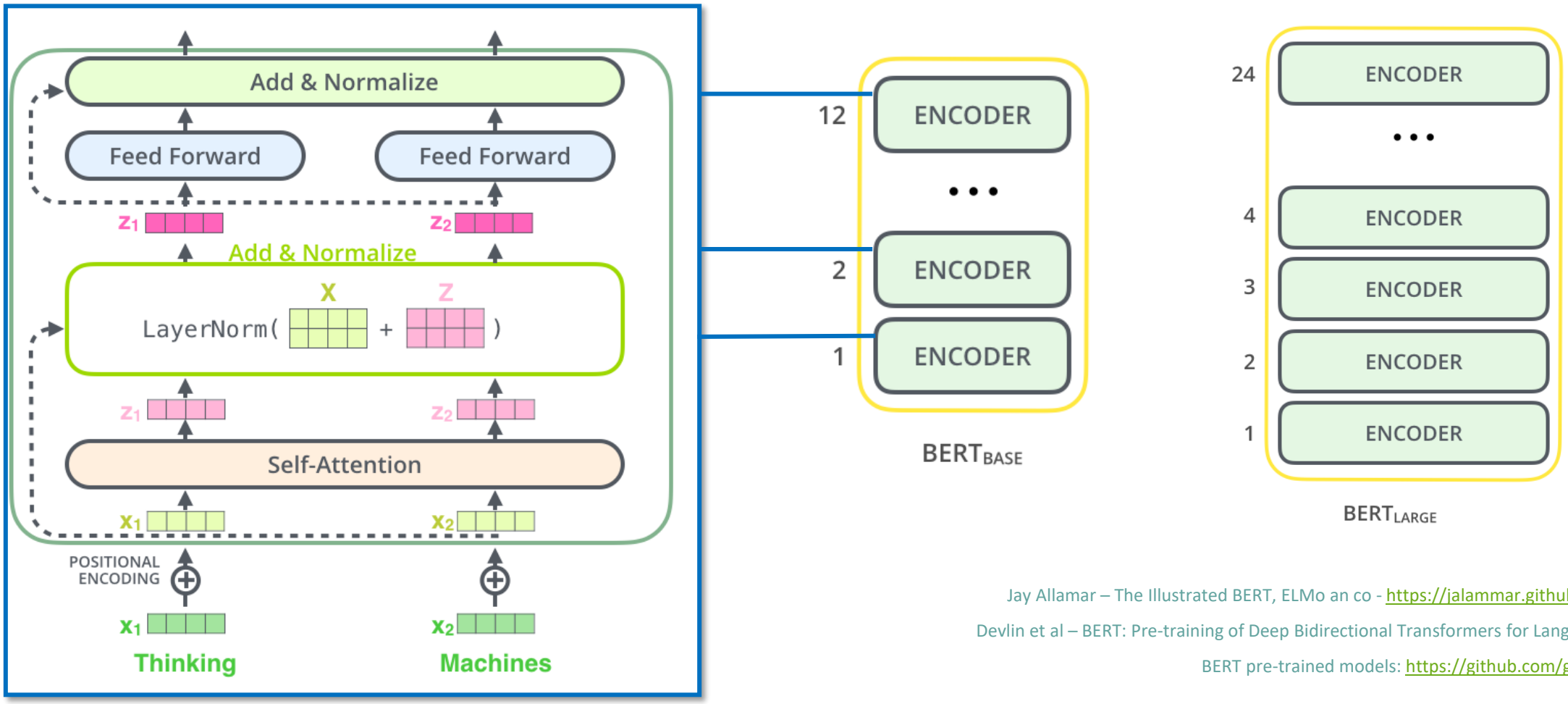
Jay Allamar – The Illustrated Transformer - https://jalammar.github.io/illustrated-transformer/

Vaswani et al – Attention is All You need

Ba et Al – Layer Normalization

Large Language models: 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/

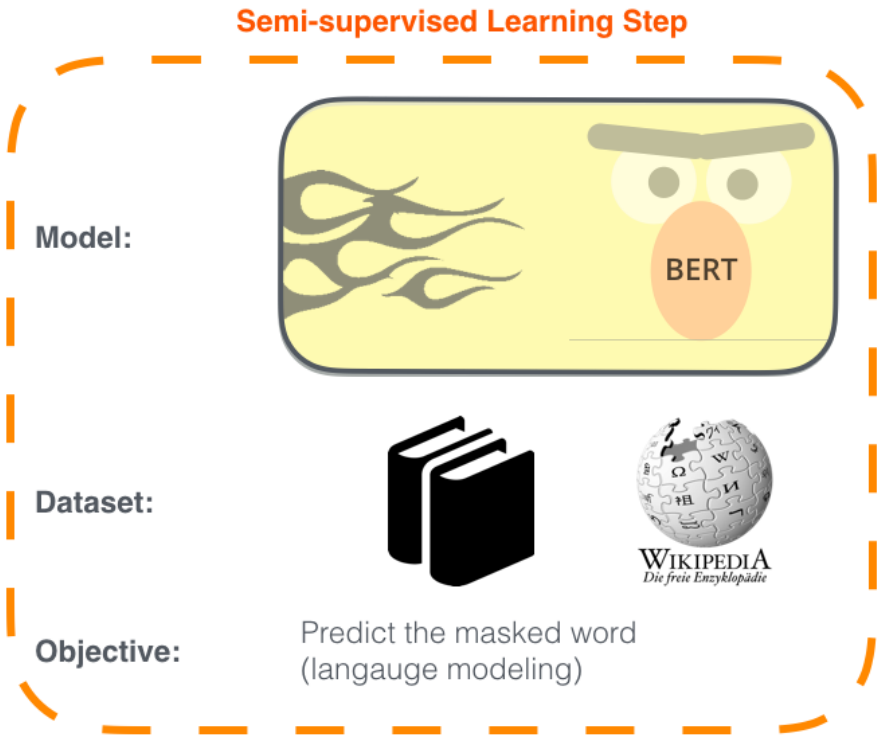
Devlin et al – BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding

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

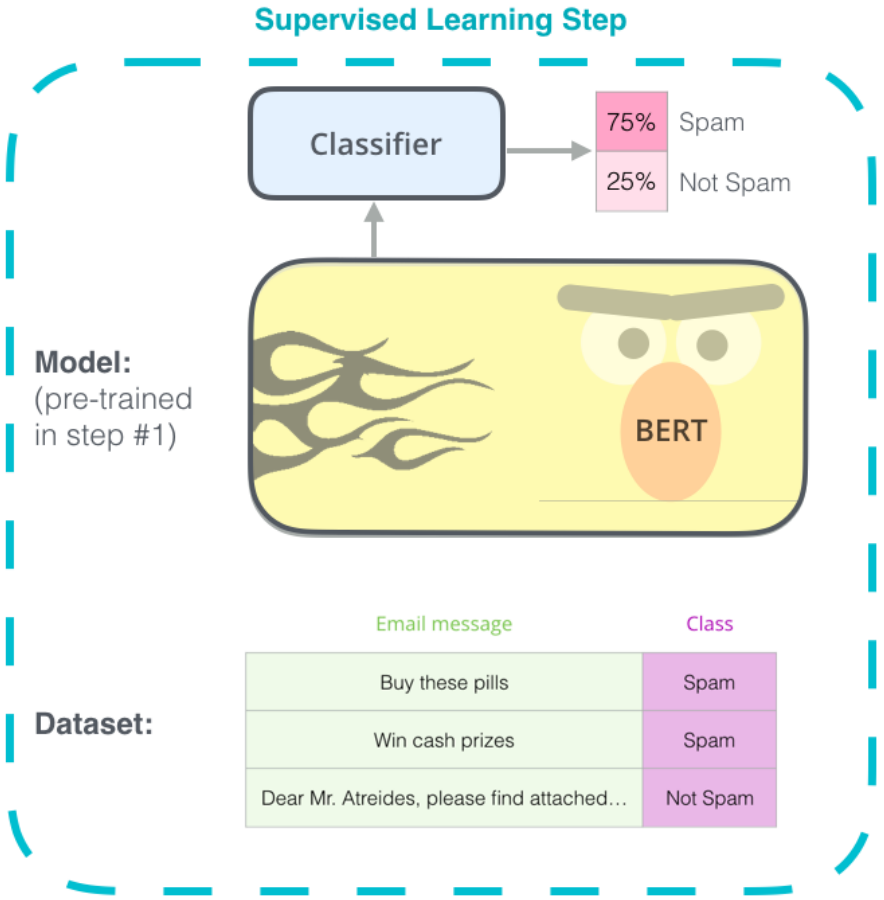
BERT usage

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.

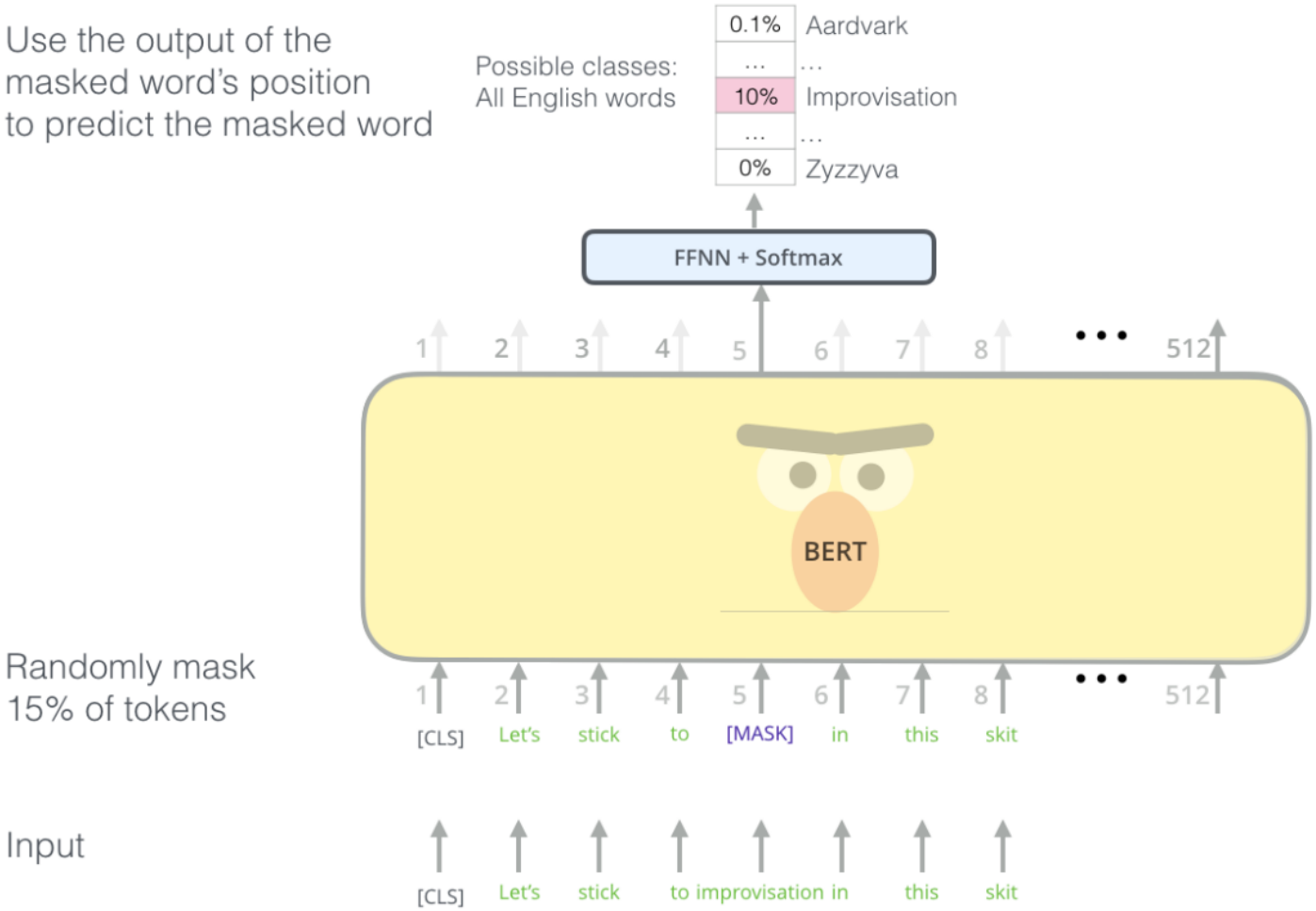


Jay Allamar – The Illustrated BERT, ELMo an co - https://jalammar.github.io/illustrated-bert/

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

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

BERT pretraining



Task #1: Masked Language Model

Randomly replace 15% of tokens in the text by the special [MASK] token.

Append a "language model head" to the output of the last BERT hidden layer:

- Fully connected layer + activation + layer normalization
- Fully connected layer: can be transpose of token embeddings matrix + bias
- Softmax over all vocabulary words

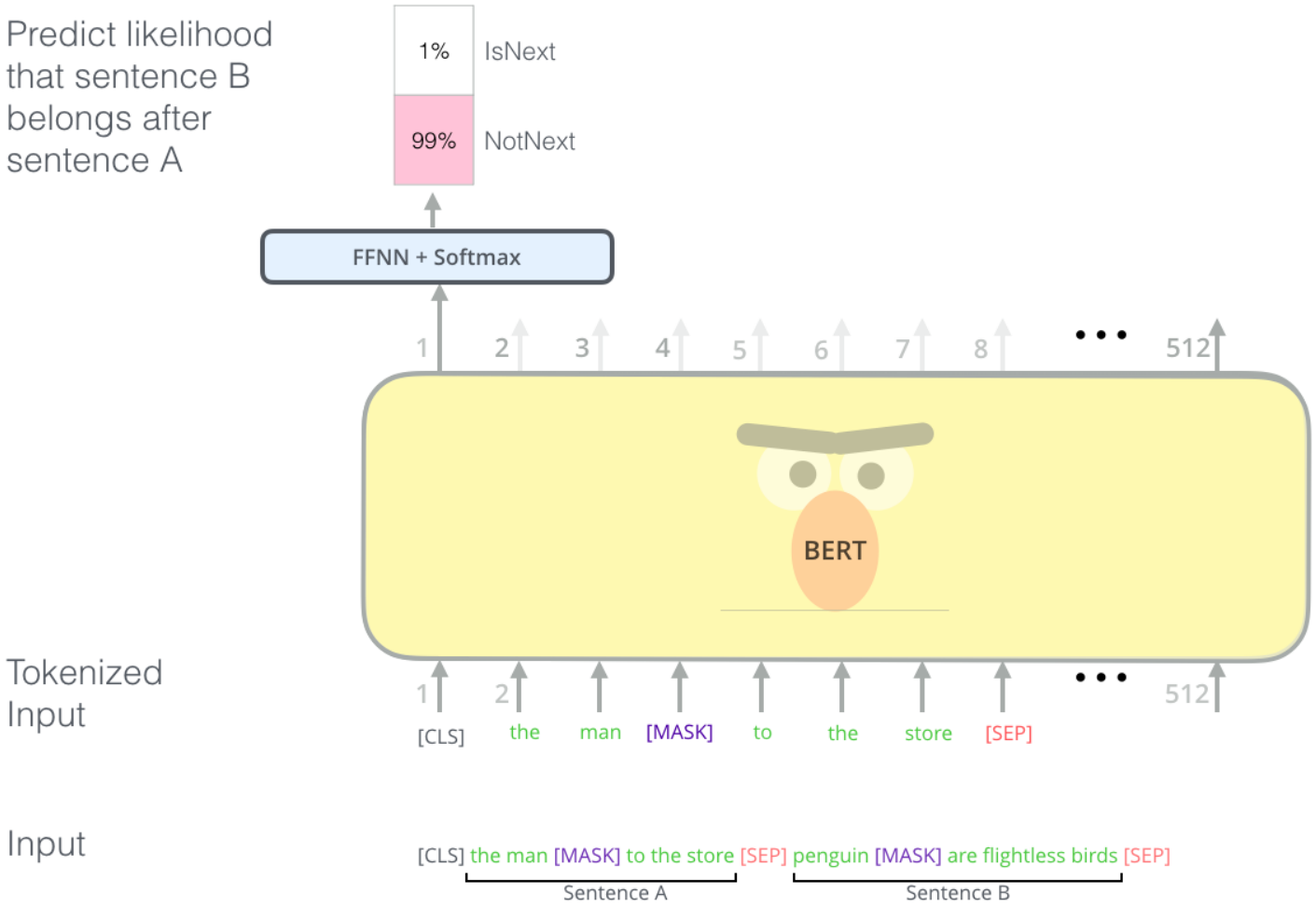
Training tries to minimize cross-entropy loss of predicting the original word at [MASK].

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

BERT pretraining



Task #2: Next Sentence Prediction

Choose sentences A and B from the training data.

- 50% of the times B follows A in the text
- 50% of the times B is a random sentence from the corpus.

Predict whether B follows A.

Inputs: [CLS] + A + [SEP] + B + [SEP]

Append a "next sentence prediction" head to the final hidden values of the [CLS] token:

- Fully connected layer + softmax over two outputs (true/false)

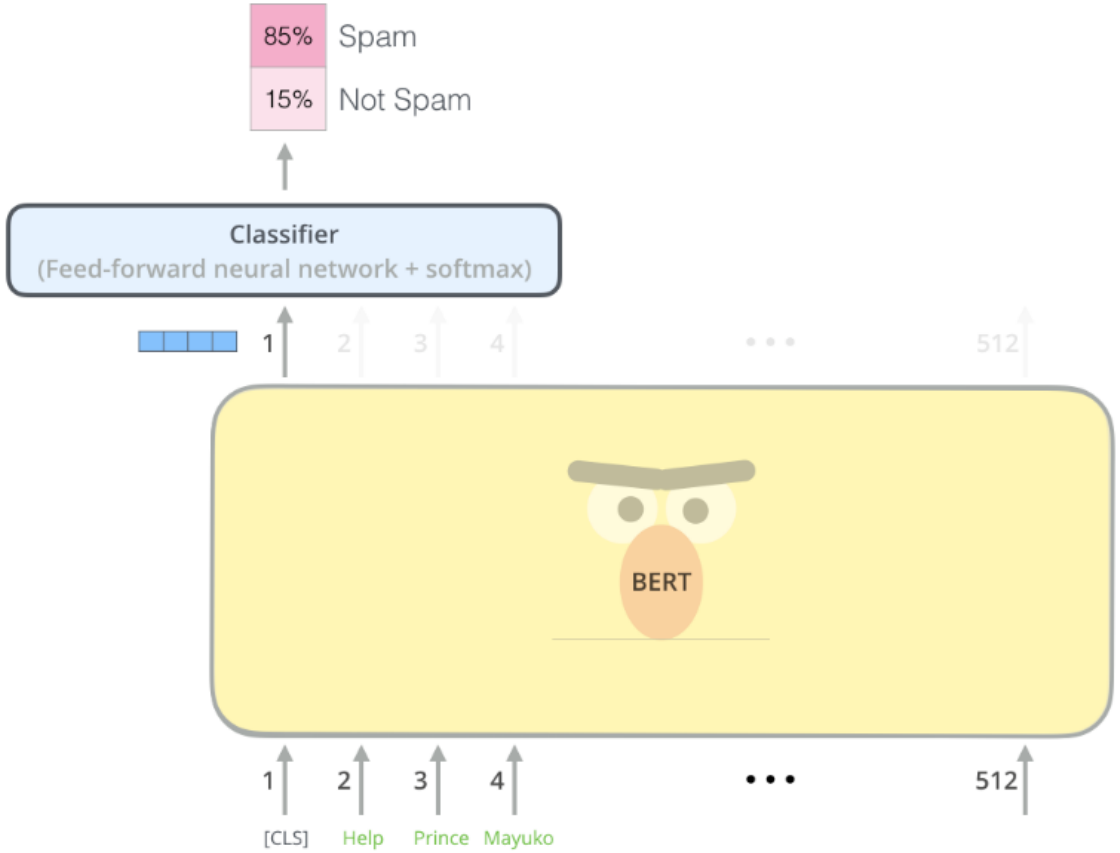
Jay Allamar – The Illustrated BERT, ELMo an co - https://jalammar.github.io/illustrated-bert/

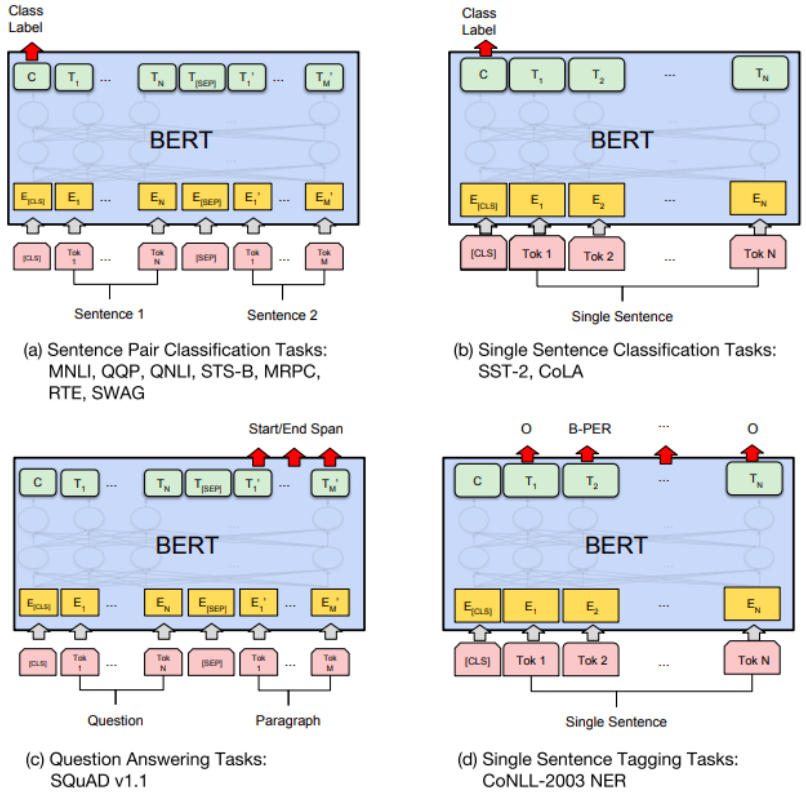
Devlin et al – BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding

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

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.



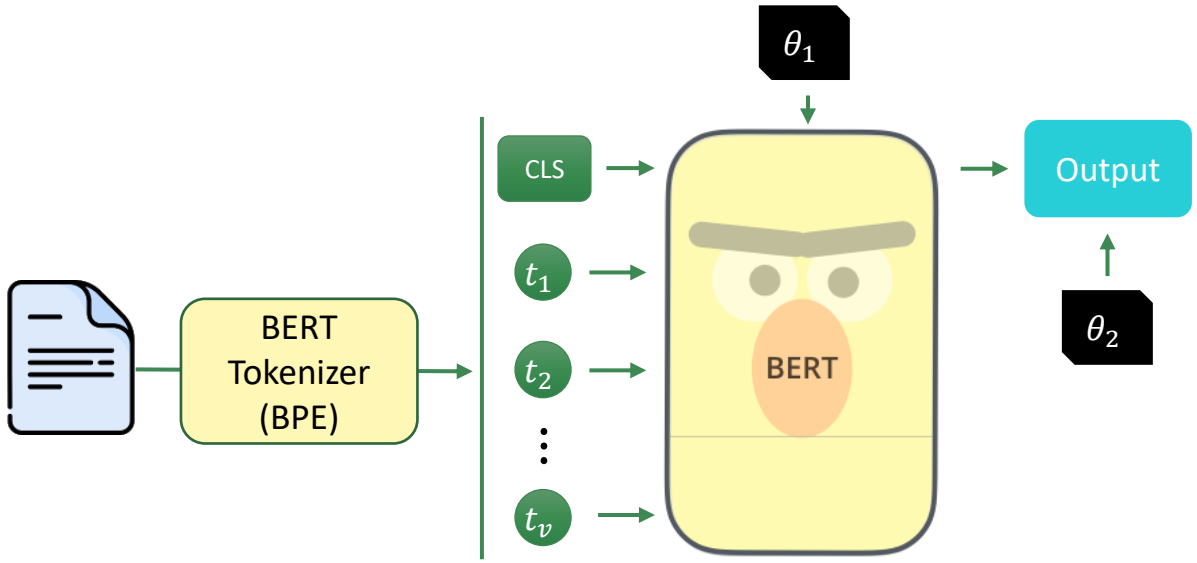


Jay Allamar – The Illustrated BERT, ELMo an co - https://jalammar.github.io/illustrated-bert/

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

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

BERT applied model



Transformers

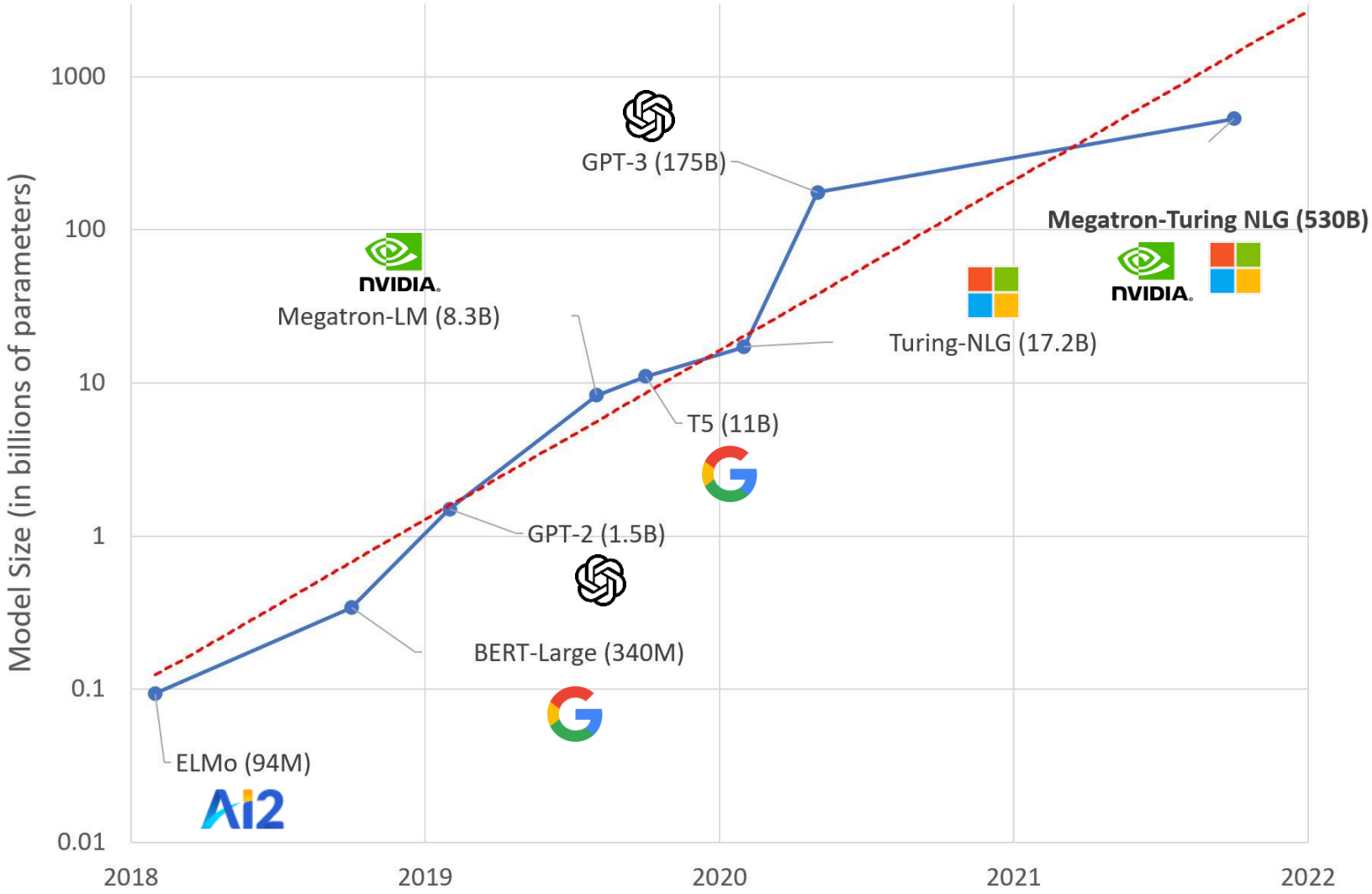
build passing | license Apache-2.0 | website online | release v2.1.1

State-of-the-art Natural Language Processing for TensorFlow 2.0 and PyTorch

🤗 Transformers (formerly known as `pytorch-transformers` and `pytorch-pretrained-bert`) provides state-of-the-art general-purpose architectures (BERT, GPT-2, RoBERTa, XLM, DistilBert, XLNet, CTRL...) for Natural Language Understanding (NLU) and Natural Language Generation (NLG) with over 32+ pretrained models in 100+ languages and deep interoperability between TensorFlow 2.0 and PyTorch.

https://github.com/huggingface/transformers

Evolution of large language models



https://www.microsoft.com/en-us/research/blog/using-deepspeed-and-megatron-to-train-megatron-turing-nlg-530b-the-worlds-largest-and-most-powerful-generative-language-model/

RoBERTa

RoBERTa proposes a better way to pre-train a BERT model, by including the following modifications

- **Dynamic masking**: resample MASK tokens at every epoch.
- **Remove the Next Sentence Prediction task**, and create input patterns for the Masked Language Task by grouping contiguous sentences from the same document (at most 512 tokens).
- **Large batches**: use a batch size of 8K, via distributed parallelization or gradient accumulation.
- **Byte-based BPE** tokenization.

Model	data	bsz	steps	SQuAD (v1.1/2.0)	MNLI-m	SST-2
RoBERTa						
with BOOKS + WIKI	16GB	8K	100K	93.6/87.3	89.0	95.3
+ additional data (§3.2)	160GB	8K	100K	94.0/87.7	89.3	95.6
+ pretrain longer	160GB	8K	300K	94.4/88.7	90.0	96.1
+ pretrain even longer	160GB	8K	500K	**94.6/89.4**	**90.2**	**96.4**
$\text{BERT}_{\text{LARGE}}$						
with BOOKS + WIKI	13GB	256	1M	90.9/81.8	86.6	93.7

Using the same pre-training dataset (BookCorpus + English Wikipedia), RoBERTa achieves better performance than standard BERT in several downstream tasks.

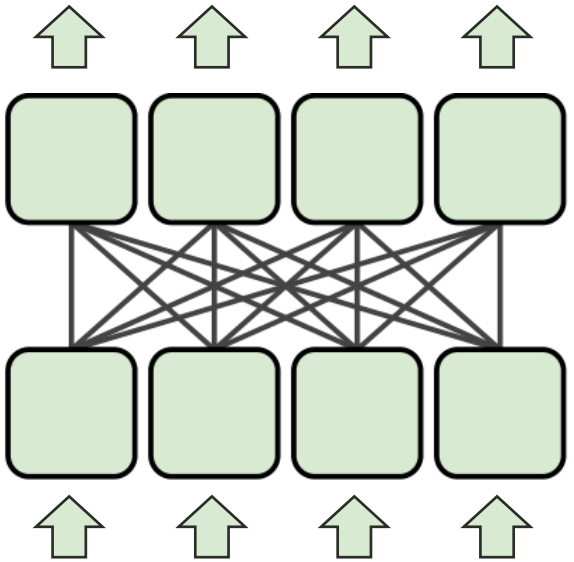
Performance is improved by adding more datasets: Common Crawl News + OpenWebText + STORIES. 160GB in total.

Even better results by pretraining more steps. No signs of overfitting.

RoBERTa: An optimized method for pretraining self-supervised NLP systems - https://ai.facebook.com/blog/roberta-an-optimized-method-for-pretraining-self-supervised-nlp-systems/

RoBERTa: A Robustly Optimized BERT Pretraining Approach - https://arxiv.org/pdf/1907.11692.pdf

Encoder and decoder architectures

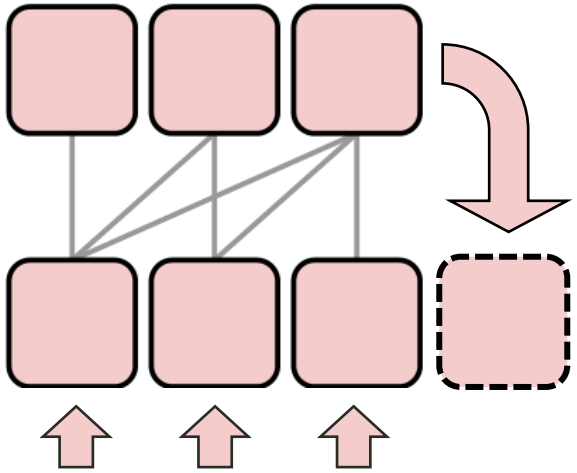


Encoder

Each token can attend to every other token in the sequence. Each token can generate an output.

Useful for document and token classification.

Architecture used by BERT, RoBERTa and other extensions (ALBERT, DeBERTa, …)



Decoder

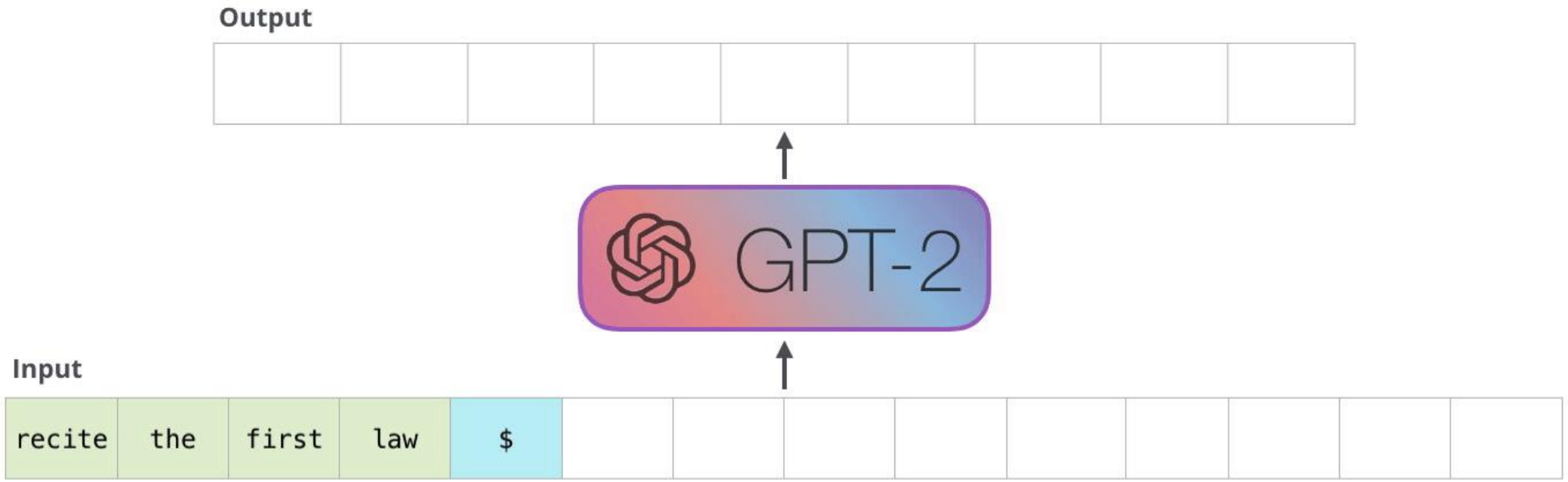
Each token can only attend previous tokens in the sequence. The last token generates an input that can be fed back as a new input.

Useful for text generation.

Architecture used by GPT.

Raffel et al - Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer - https://arxiv.org/pdf/1910.10683.pdf

Decoder architecture for text generation

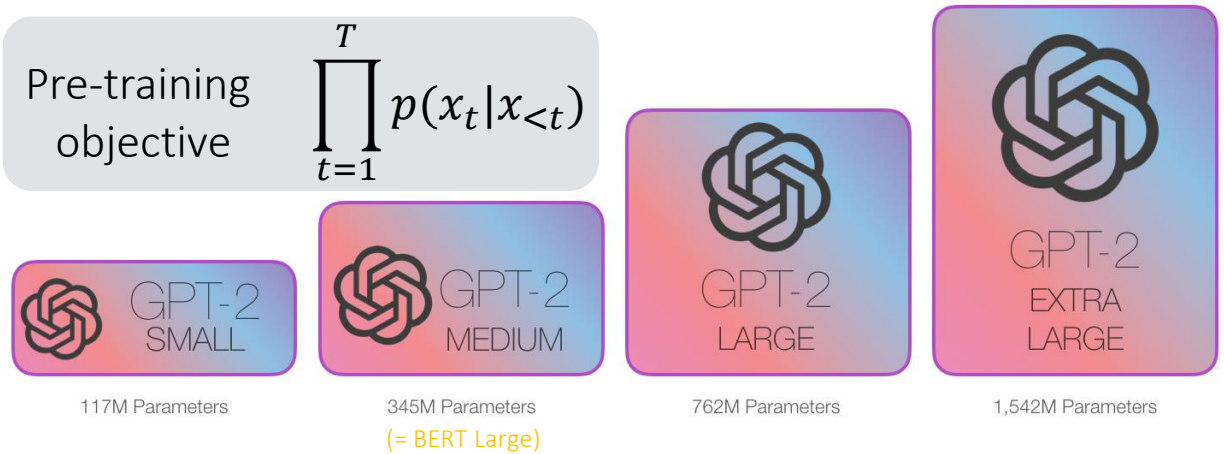


Jay Alammar - The Illustrated GPT-2 (Visualizing Transformer Language Models) - https://jalammar.github.io/illustrated-gpt2/

GPT-2

GPT-2 is an even larger Transformer model, up to 48 layers. Pre-training is done by predicting next word in the sequence, using causal masking in transformer layers (decoder architecture). Improves over BERT results.

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



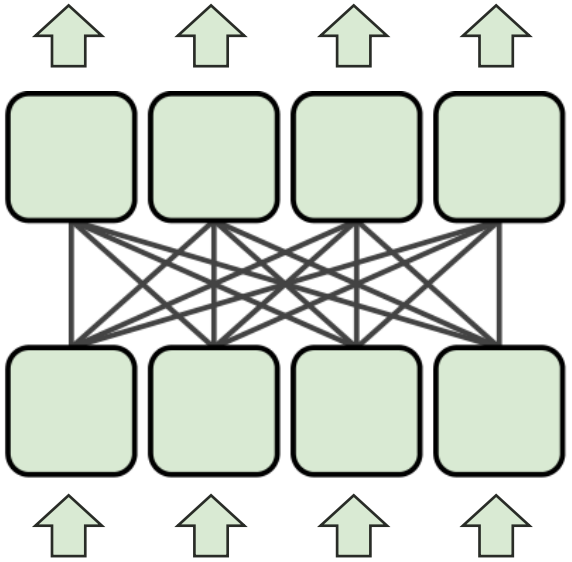
Training corpus: crawl over Reddit, selecting every link that has received at least 3 karma points, then download the pages those links point to. A fast way to obtain a curated corpus of quality language.

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

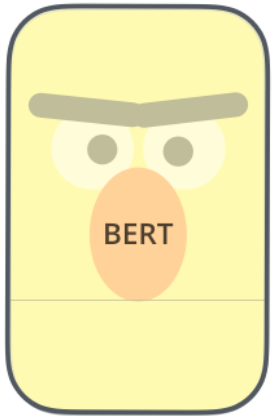
Racford et al – Language Models are Unsupervised Multitask Learners - https://cdn.openai.com/better-language-models/language_models_are_unsupervised_multitask_learners.pdf

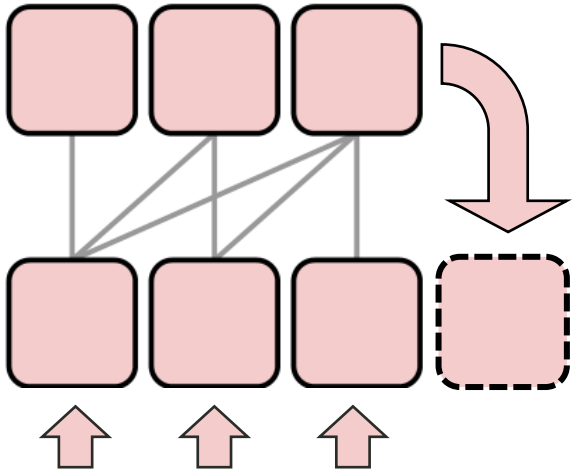
Jay Alammar - The Illustrated GPT-2 (Visualizing Transformer Language Models) - https://jalammar.github.io/illustrated-gpt2/

Encoder-Decoder architecture

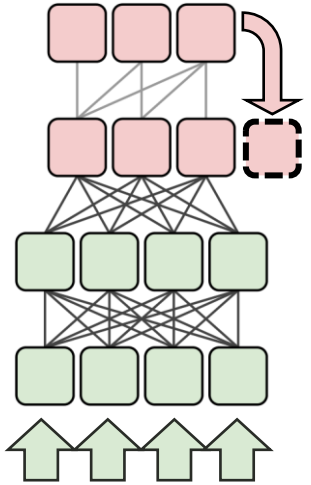


Encoder





Decoder



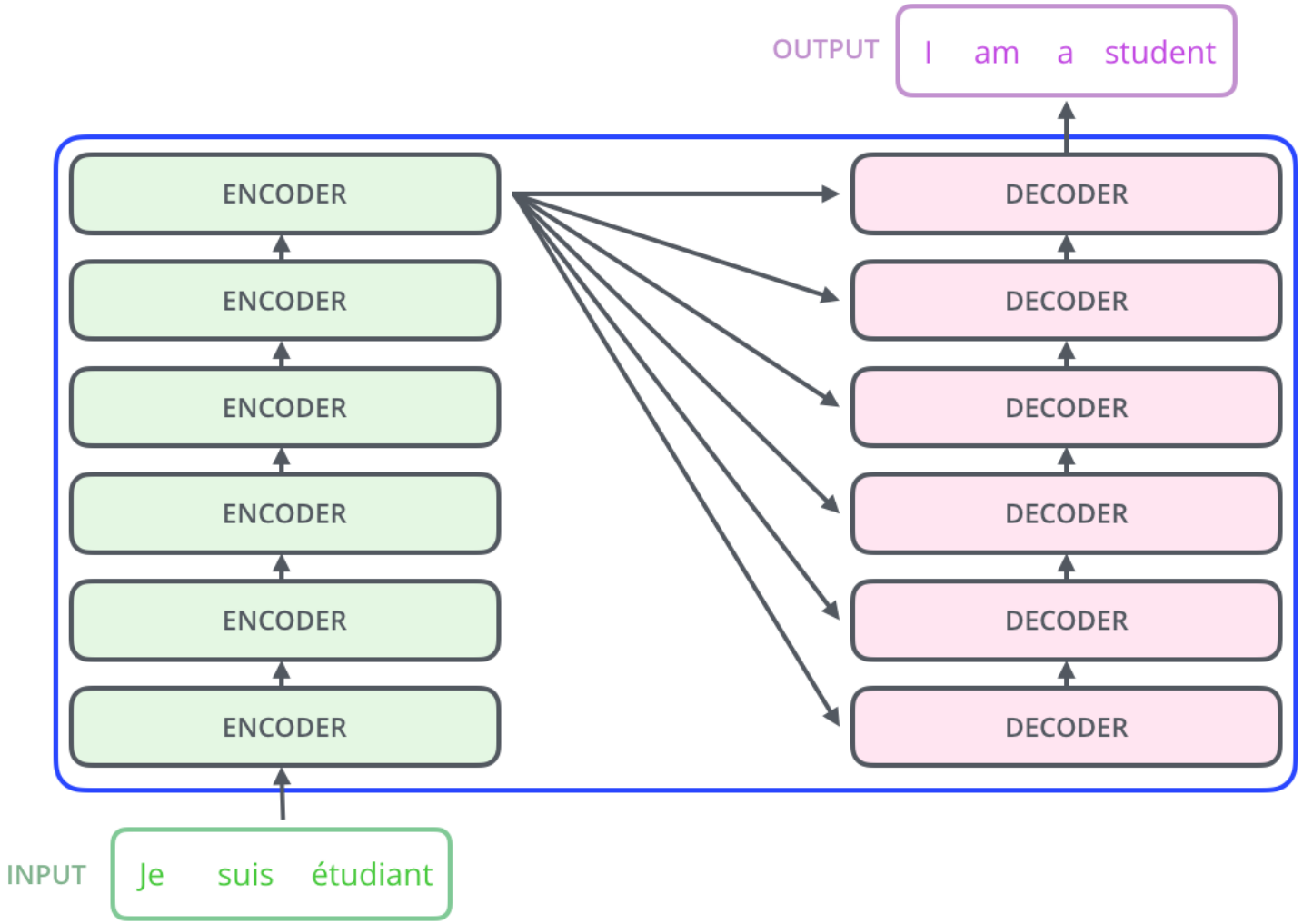
Encoder-Decoder

Each input token can attend to every other input token, but output tokens can only attend to previous output tokens.

Useful for text generation with a context (translation, question answering).

Raffel et al - Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer - https://arxiv.org/pdf/1910.10683.pdf

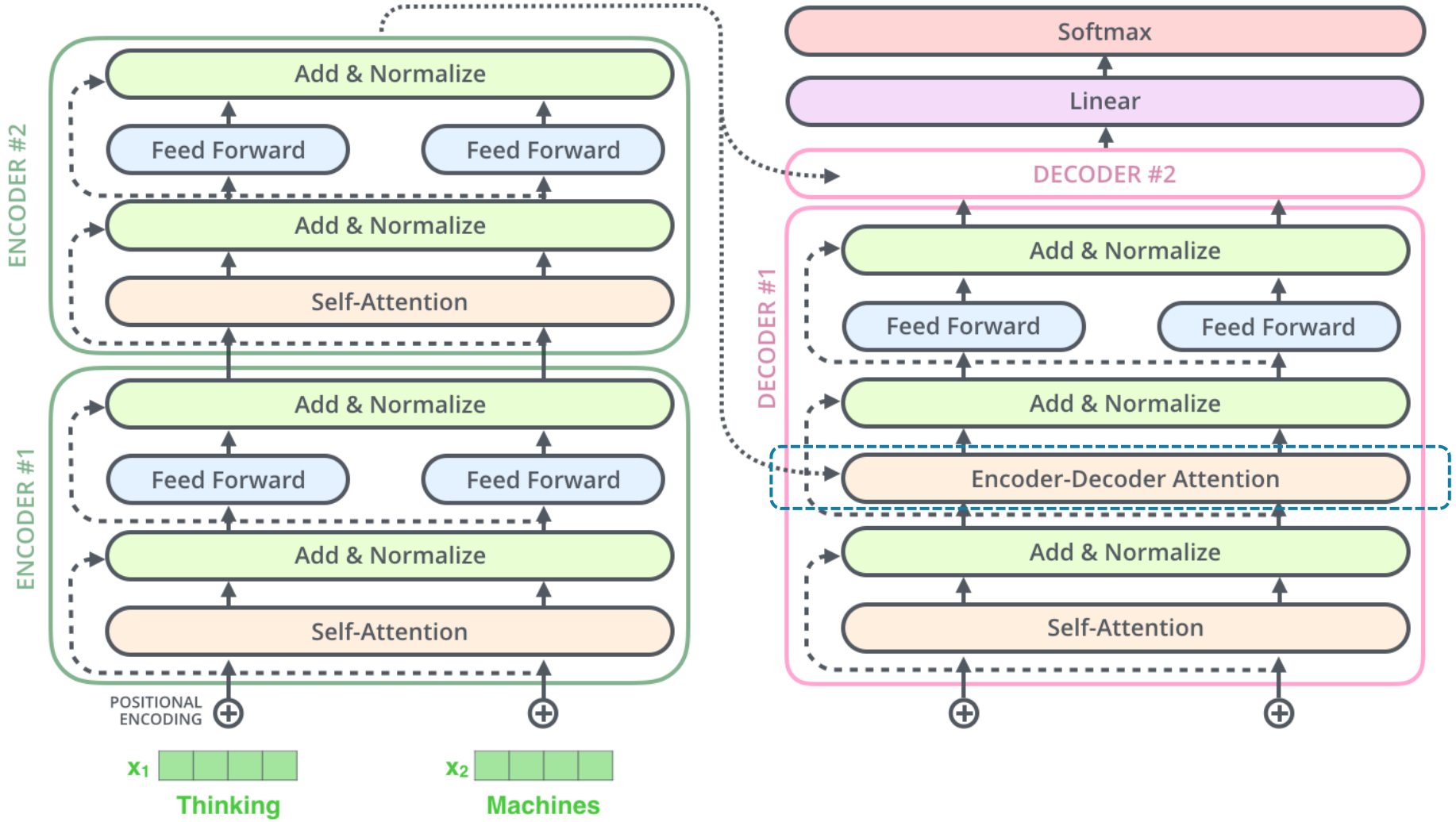
Encoder-decoder models



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

Vaswani et al – Attention is All You need

Encoder-Decoder architecture with Transformers

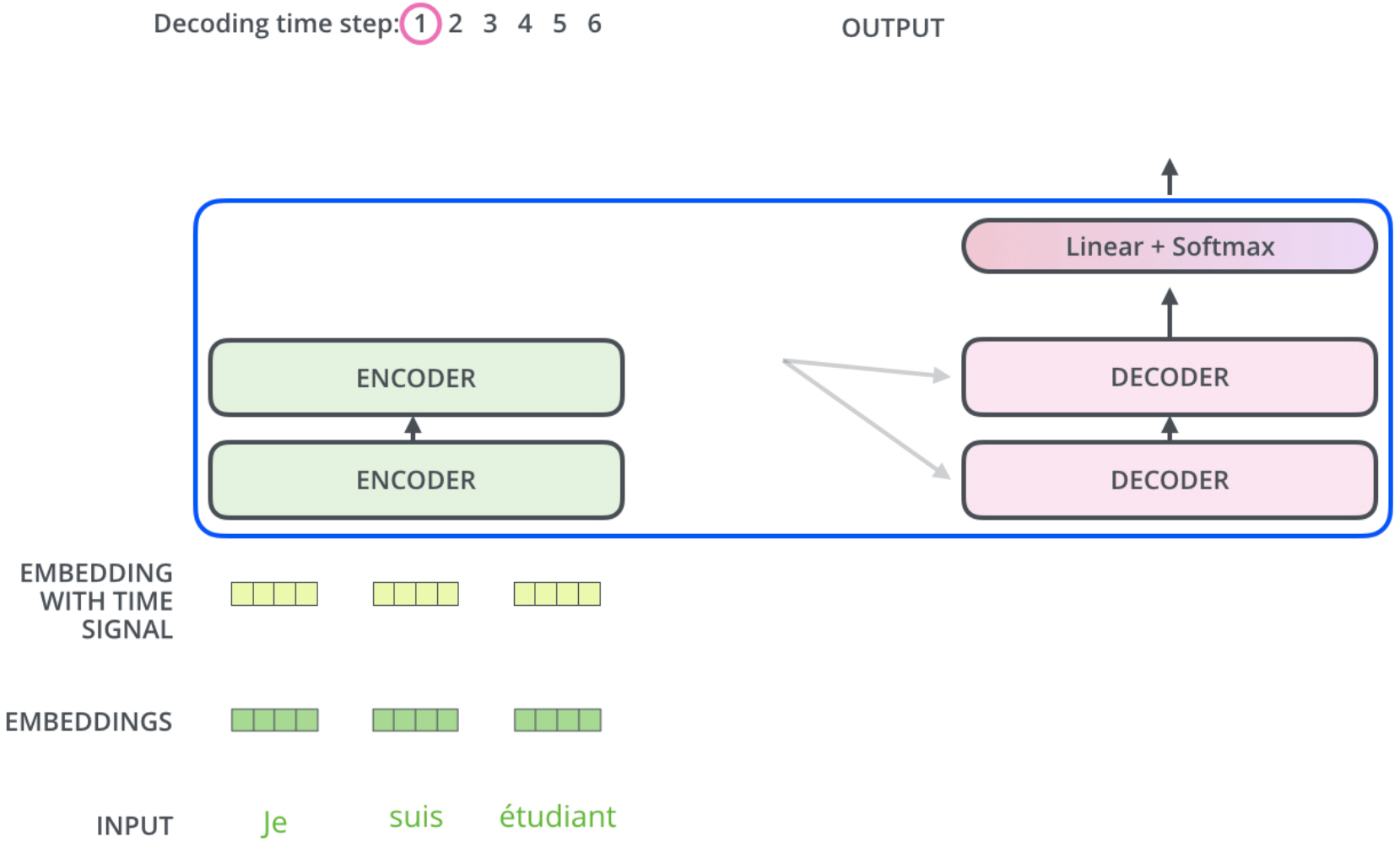


Mix encoder outputs with translation so far

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

Vaswani et al – Attention is All You need

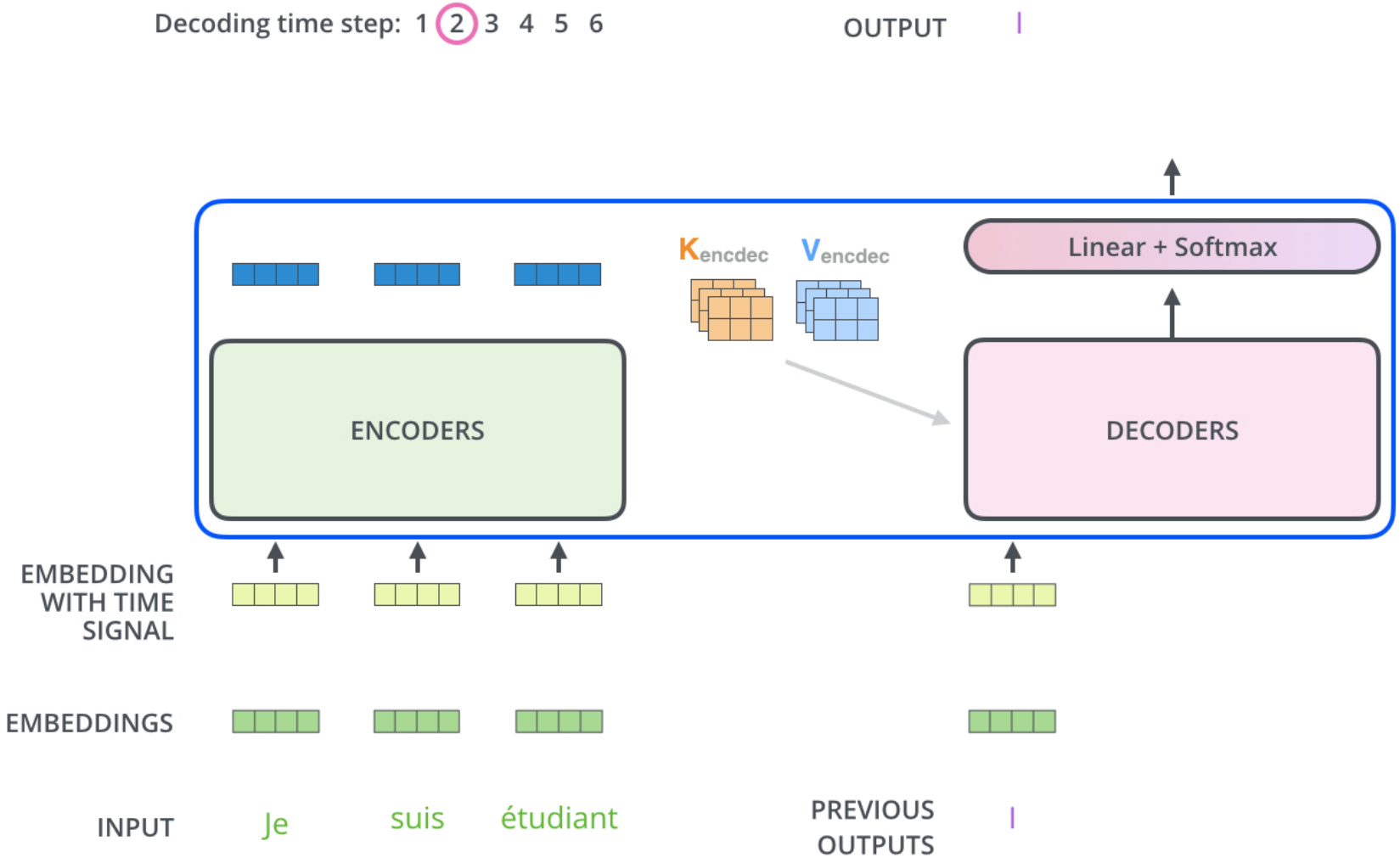
Encoder-Decoder architecture: first decoding step



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

Vaswani et al – Attention is All You need

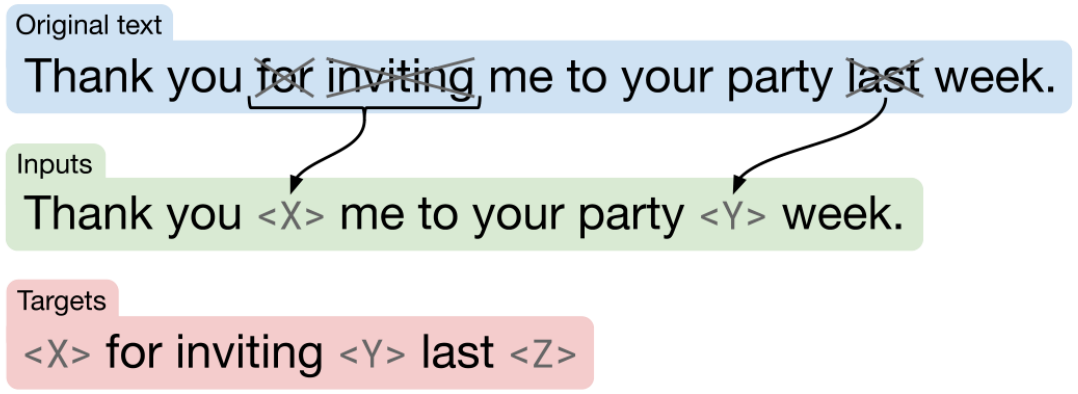
Encoder-Decoder architecture: next decoding steps



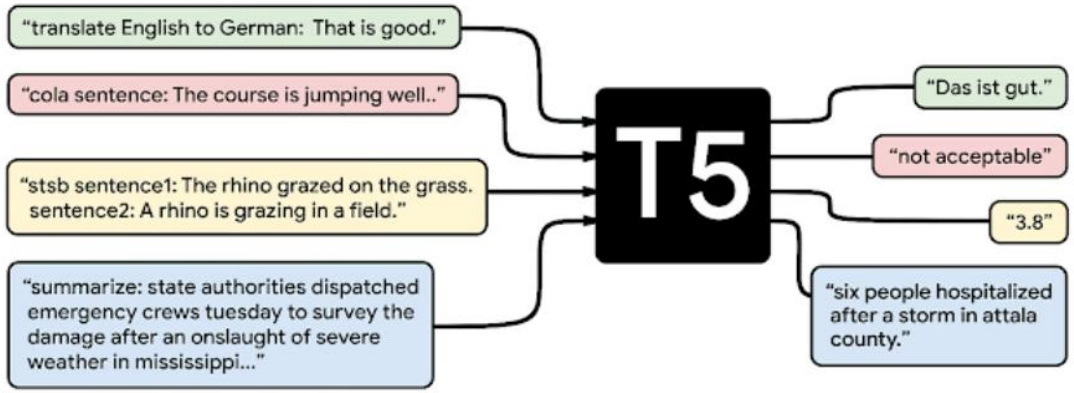
Jay Allamar – The Illustrated Transformer - https://jalammar.github.io/illustrated-transformer/

Vaswani et al – Attention is All You need

Text-To-Text Transfer Transformer (T5)



T5 is an encoder-decored model that unifies a variety of NLP tasks by casting all of them as a text-to-text problems. The language model is pretrained to generate text to fill one or more gaps in the input text. The Colossal Clean Crawled Corpus (C4), a cleaned version of Common Crawl, is used for pretraining.



In the fine-tuning phase, T5 can be adapted to any task that receives text and generates text as an answer. Even classification or regression tasks can be framed as generating a text with the name of the class or the number to predict.

Raffel et al - Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer - https://arxiv.org/pdf/1910.10683.pdf

Exploring Transfer Learning with T5: the Text-To-Text Transfer Transformer - https://ai.googleblog.com/2020/02/exploring-transfer-learning-with-t5.html

GPT-3

Language Models are Few-Shot Learners

Tom B. Brown* **Benjamin Mann*** **Nick Ryder*** **Melanie Subbiah***

Jared Kaplan† **Prafulla Dhariwal** **Arvind Neelakantan** **Pranav Shyam** **Girish Sastry**

Amanda Askell **Sandhini Agarwal** **Ariel Herbert-Voss** **Gretchen Krueger** **Tom Henighan**

Rewon Child **Aditya Ramesh** **Daniel M. Ziegler** **Jeffrey Wu** **Clemens Winter**

Christopher Hesse **Mark Chen** **Eric Sigler** **Mateusz Litwin** **Scott Gray**

Benjamin Chess **Jack Clark** **Christopher Berner**

Sam McCandlish **Alec Radford** **Ilya Sutskever** **Dario Amodei**

OpenAI

Abstract

Recent work has demonstrated substantial gains on many NLP tasks and benchmarks by pre-training on a large corpus of text followed by fine-tuning on a specific task. While typically task-agnostic in architecture, this method still requires task-specific fine-tuning datasets of thousands or tens of thousands of examples. By contrast, humans can generally perform a new language task from only a few examples or from simple instructions – something which current NLP systems still largely struggle to do. Here we show that scaling up language models greatly improves task-agnostic, few-shot performance, sometimes even reaching competitiveness with prior state-of-the-art fine-tuning approaches. Specifically, we train GPT-3, an autoregressive language model with 175 billion parameters, 10x more than any previous non-sparse language model, and test its performance in the few-shot setting. For all tasks, GPT-3 is applied without any gradient updates or fine-tuning, with tasks and few-shot demonstrations specified purely via text interaction with the model. GPT-3 achieves strong performance on many NLP datasets, including translation, question-answering, and cloze tasks, as well as several tasks that require on-the-fly reasoning or domain adaptation, such as unscrambling words, using a novel word in a sentence, or performing 3-digit arithmetic. At the same time, we also identify some datasets where GPT-3's few-shot learning still struggles, as well as some datasets where GPT-3 faces methodological issues related to training on large web corpora. Finally, we find that GPT-3 can generate samples of news articles which human evaluators have difficulty distinguishing from articles written by humans. We discuss broader societal impacts of this finding and of GPT-3 in general.

GPT-3

The three settings we explore for in-context learning

Zero-shot

The model predicts the answer given only a natural language description of the task. No gradient updates are performed.

```
1  Translate English to French:        ← task description
2  cheese =>                           ← prompt
```

One-shot

In addition to the task description, the model sees a single example of the task. No gradient updates are performed.

```
1  Translate English to French:        ← task description
2  sea otter => loutre de mer          ← example
3  cheese =>                           ← prompt
```

Few-shot

In addition to the task description, the model sees a few examples of the task. No gradient updates are performed.

```
1  Translate English to French:        ← task description
2  sea otter => loutre de mer          ← examples
3  peppermint => menthe poivrée        ←
4  plush girafe => girafe peluche      ←
5  cheese =>                           ← prompt
```

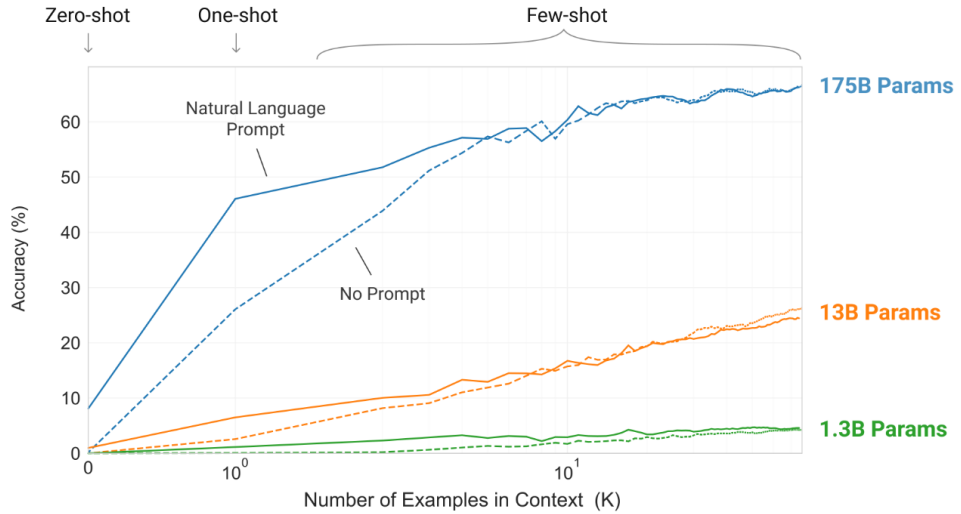


Figure 1.2: Larger models make increasingly efficient use of in-context information. We show in-context learning performance on a simple task requiring the model to remove random symbols from a word, both with and without a natural language task description (see Sec. 3.9.2). The steeper "in-context learning curves" for large models demonstrate improved ability to learn a task from contextual information. We see qualitatively similar behavior across a wide range of tasks.

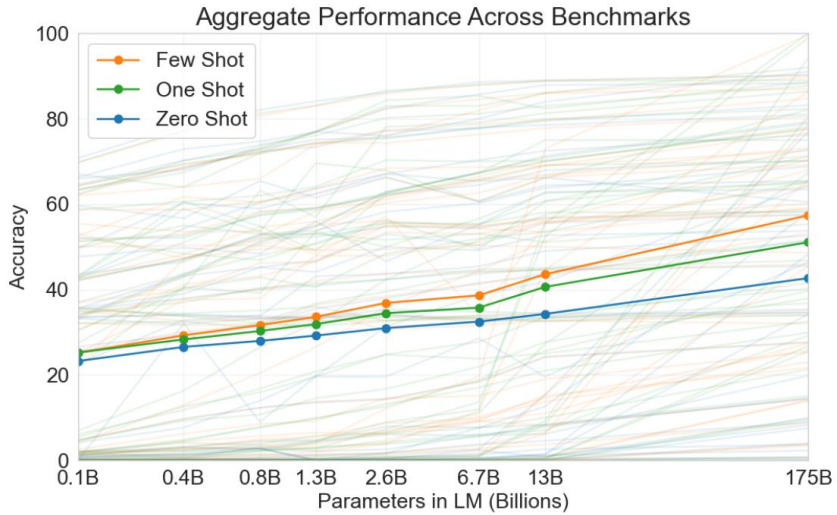


Figure 1.3: Aggregate performance for all 42 accuracy-denominated benchmarks While zero-shot performance improves steadily with model size, few-shot performance increases more rapidly, demonstrating that larger models are more proficient at in-context learning. See Figure 3.8 for a more detailed analysis on SuperGLUE, a standard NLP benchmark suite.

Language Models are Few-Shot Learners - https://arxiv.org/abs/2005.14165

GPT-3

Figure 3.5: Zero-, one-, and few-shot performance on the adversarial Winogrande dataset as model capacity scales. Scaling is relatively smooth with the gains to few-shot learning increasing with model size, and few-shot GPT-3 175B is competitive with a fine-tuned RoBERTA-large.

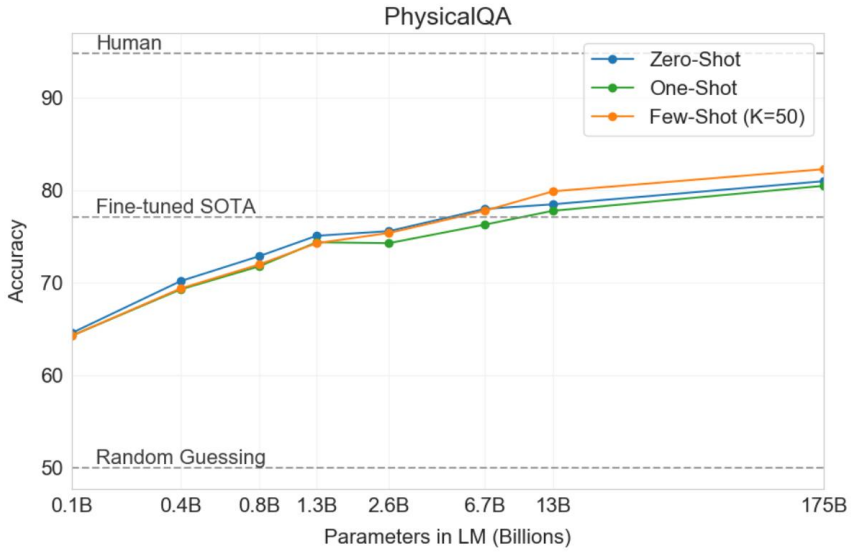


Figure 3.6: GPT-3 results on PIQA in the zero-shot, one-shot, and few-shot settings. The largest model achieves a score on the development set in all three conditions that exceeds the best recorded score on the task.

Language Models are Few-Shot Learners - https://arxiv.org/abs/2005.14165

GPT-3

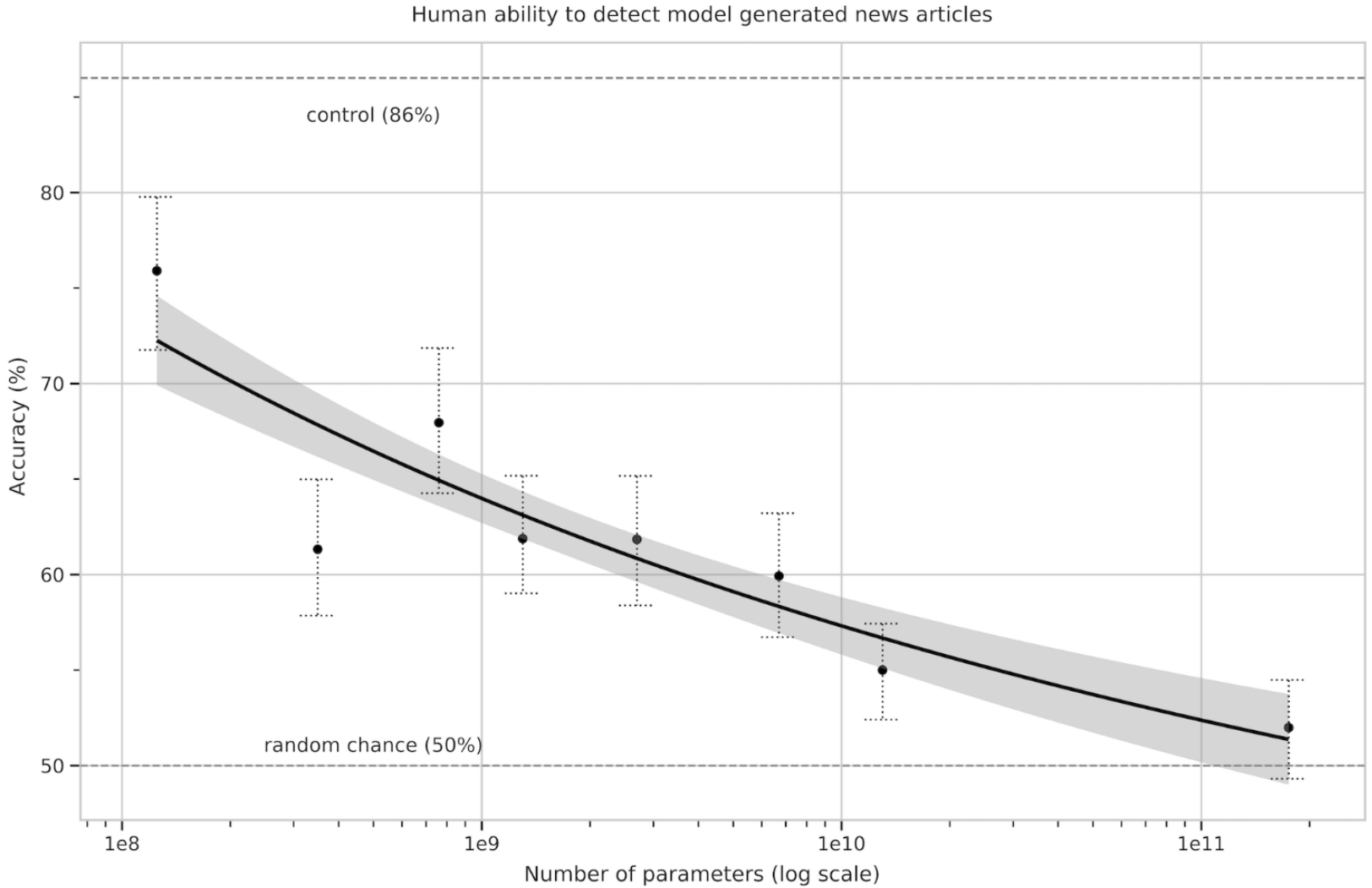


Figure 3.13: People's ability to identify whether news articles are model-generated (measured by the ratio of correct assignments to non-neutral assignments) decreases as model size increases. Accuracy on the outputs on the deliberately-bad control model (an unconditioned GPT-3 Small model with higher output randomness) is indicated with the dashed line at the top, and the random chance (50%) is indicated with the dashed line at the bottom. Line of best fit is a power law with 95% confidence intervals.

Language Models are Few-Shot Learners - https://arxiv.org/abs/2005.14165

GPT-3

Title: United Methodists Agree to Historic Split
Subtitle: Those who oppose gay marriage will form their own denomination
Article: After two days of intense debate, the United Methodist Church has agreed to a historic split - one that is expected to end in the creation of a new denomination, one that will be "theologically and socially conservative," according to The Washington Post. The majority of delegates attending the church's annual General Conference in May voted to strengthen a ban on the ordination of LGBTQ clergy and to write new rules that will "discipline" clergy who officiate at same-sex weddings. But those who opposed these measures have a new plan: They say they will form a separate denomination by 2020, calling their church the Christian Methodist denomination.
The Post notes that the denomination, which claims 12.5 million members, was in the early 20th century the "largest Protestant denomination in the U.S.," but that it has been shrinking in recent decades. The new split will be the second in the church's history. The first occurred in 1968, when roughly 10 percent of the denomination left to form the Evangelical United Brethren Church. The Post notes that the proposed split "comes at a critical time for the church, which has been losing members for years," which has been "pushed toward the brink of a schism over the role of LGBTQ people in the church." Gay marriage is not the only issue that has divided the church. In 2016, the denomination was split over ordination of transgender clergy, with the North Pacific regional conference voting to ban them from serving as clergy, and the South Pacific regional conference voting to allow them.

Figure 3.14: The GPT-3 generated news article that humans had the greatest difficulty distinguishing from a human written article (accuracy: 12%).

Language Models are Few-Shot Learners - https://arxiv.org/abs/2005.14165

GPT-3

A "whatpu" is a small, furry animal native to Tanzania. An example of a sentence that uses the word whatpu is:
We were traveling in Africa and we saw these very cute whatpus.

To do a "farduddle" means to jump up and down really fast. An example of a sentence that uses the word farduddle is:
One day when I was playing tag with my little sister, she got really excited and she started doing these crazy farduddles.
A "yalubalu" is a type of vegetable that looks like a big pumpkin. An example of a sentence that uses the word yalubalu is:
I was on a trip to Africa and I tried this yalubalu vegetable that was grown in a garden there. It was delicious.

A "Burringo" is a car with very fast acceleration. An example of a sentence that uses the word Burringo is:
In our garage we have a Burringo that my father drives to work every day.

A "Gigamuru" is a type of Japanese musical instrument. An example of a sentence that uses the word Gigamuru is:
I have a Gigamuru that my uncle gave me as a gift. I love to play it at home.

To "screeg" something is to swing a sword at it. An example of a sentence that uses the word screeg is:
We screeghed at each other for several minutes and then we went outside and ate ice cream.

Figure 3.16: Representative GPT-3 completions for the few-shot task of using a new word in a sentence. Boldface is GPT-3's completions, plain text is human prompts. In the first example both the prompt and the completion are provided by a human; this then serves as conditioning for subsequent examples where GPT-3 receives successive additional prompts and provides the completions. Nothing task-specific is provided to GPT-3 other than the conditioning shown here.

Language Models are Few-Shot Learners - https://arxiv.org/abs/2005.14165

GPT-3

OpenAI Playground: https://beta.openai.com/playground

Transformers pre-trained models demo

BERT-like models in other languages

The community has created other models following BERT and RoBERTa pre-training strategies, for many other languages apart from english. Some amusing examples:

CamemBERT (french)

RobBERT (dutch)

GreekBERT (greek)

CamemBERT - https://camembert-model.fr/

RobBERT - https://github.com/iPieter/RobBERT

GreekBERT - https://github.com/nlpaueb/greek-bert

Modelos BERT en español

BETO

BERTÍN

MarIA

RigoBERTa

Cañete et al – Spanish Pre-trained BERT Model and Evaluation data - https://users.dcc.uchile.cl/~jperez/papers/pml4dc2020.pdf
MarIA spanish Language Models - https://github.com/PlanTL-GOB-ES/lm-spanish
BERTIN project - https://huggingface.co/bertin-project/bertin-roberta-base-spanish
RigoBERTa - https://www.iic.uam.es/inteligencia-artificial/procesamiento-del-lenguaje-natural/modelo-lenguaje-espanol-rigoberta/

Modelos BERT en español - BETO

Modelo:

- BERT base
- Enmascarado tipo RoBERTa.
- Entrenado en TPU v3-8.

dcc CIENCIAS DE LA COMPUTACIÓN UNIVERSIDAD DE CHILE

Corpus:

Spanish Unnanotated Corpora (SUC)

3 mil millones de palabras

- Spanish Wikipedia, wikibooks, wikiquotes, …
- OpenSubtitles
- Europarl
- EUBookshop
- GlobalVoices
- …

SUC (Spanish Unnanotaded Corpora) - https://github.com/josecannete/spanish-corpora

Cañete et al – Spanish Pre-trained BERT Model and Evaluation data - https://users.dcc.uchile.cl/~jperez/papers/pml4dc2020.pdf

Modelos BERT en español - MarIA

Modelo:

- Iniciativa del Plan de Impulso de las Tecnologías del Lenguaje, con participación del Barcelona Supercomputing Center y la Biblioteca Nacional de España.
- Diversos modelos: RoBERTa-base, RoBERTa-large, GPT-2
- Entrenado durante 1 época: 48 horas en 64 GPUs V100 16GB RAM

Corpus:

Web crawls de la **Biblioteca Nacional de España**

-Desde 2009 a 2019

-59 TBs en bruto

-Filtro por idioma español, deduplicación, heurísticas

-570GB de datos tras limpieza

MarIA spanish Language Models - https://github.com/PlanTL-GOB-ES/lm-spanish

Modelos BERT en español - RigoBERTa

Modelo:

- DeBERTa base
- Entrenado en 8 GPUs A100
- Gold standard de validación

Corpus:

149GB de crawls de webs en español.
26 mil millones de palabras.

Porción española del corpus mC4. 433 mil millones de palabras.

69GB de noticias de medios de prensa en español.

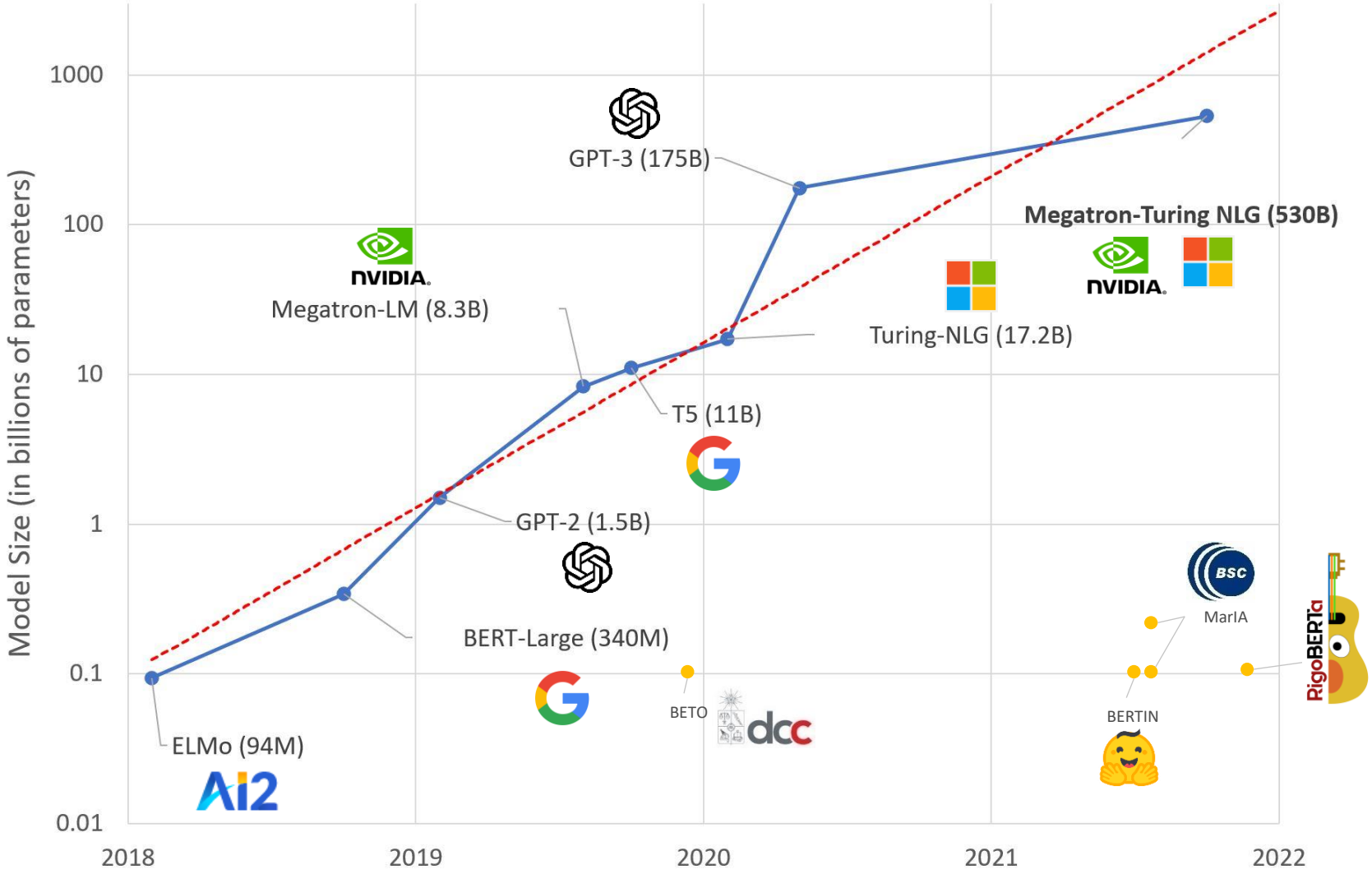
Spanish Unnanotated Corpora.
3 mil millones de palabras

RigoBERTa - https://www.iic.uam.es/inteligencia-artificial/procesamiento-del-lenguaje-natural/modelo-lenguaje-espanol-rigoberta/

RigoBERTa – Resultados del benchmark

	Dataset	BETO	BERTIN	MarIA	RigoBERTa
NER	CANTEMISTNER	89.9%	79.5%	92.3%	**93.3%** ★
NER	CAPITEL	87.0%	86.5%	**87.8%** ★	87.4%
NER	CONLL2002	89.6%	**90.1%** ★	89.9%	89.5%
Anonymize	MEDDOCAN	84.7%	72.2%	84.1%	**85.0%** ★
NER	MEDDOPROF1	80.5%	71.0%	80.7%	**83.1%** ★
NER	MEDDOPROF2	81.8%	44.2%	78.5%	**86.4%** ★
Classification	MLDOC	95.4%	94.4%	**95.6%** ★	**95.6%** ★
Paraphrasing	PAWS-X	89.7%	90.1%	88.9%	**91.0%** ★
NER	PHARMACONER	61.4%	47.1%	57.1%	**70.0%** ★
QA	SQAC	76.2%	75.0%	86.6%	**89.7%** ★
QA	SQUADES	75.6%	70.0%	81.8%	**85.4%** ★
Sentiment	TASS2020	46.1%	46.1%	**47.3%** ★	46.7%
Entailment	XNLI	81.7%	79.4%	81.6%	**83.4%** ★
	TOTALS	76.5%	69.6% x1	77.3% x3	**79.8%** ★ x10

Evolution of spanish language models



https://www.microsoft.com/en-us/research/blog/using-deepspeed-and-megatron-to-train-megatron-turing-nlg-530b-the-worlds-largest-and-most-powerful-generative-language-model/

The future

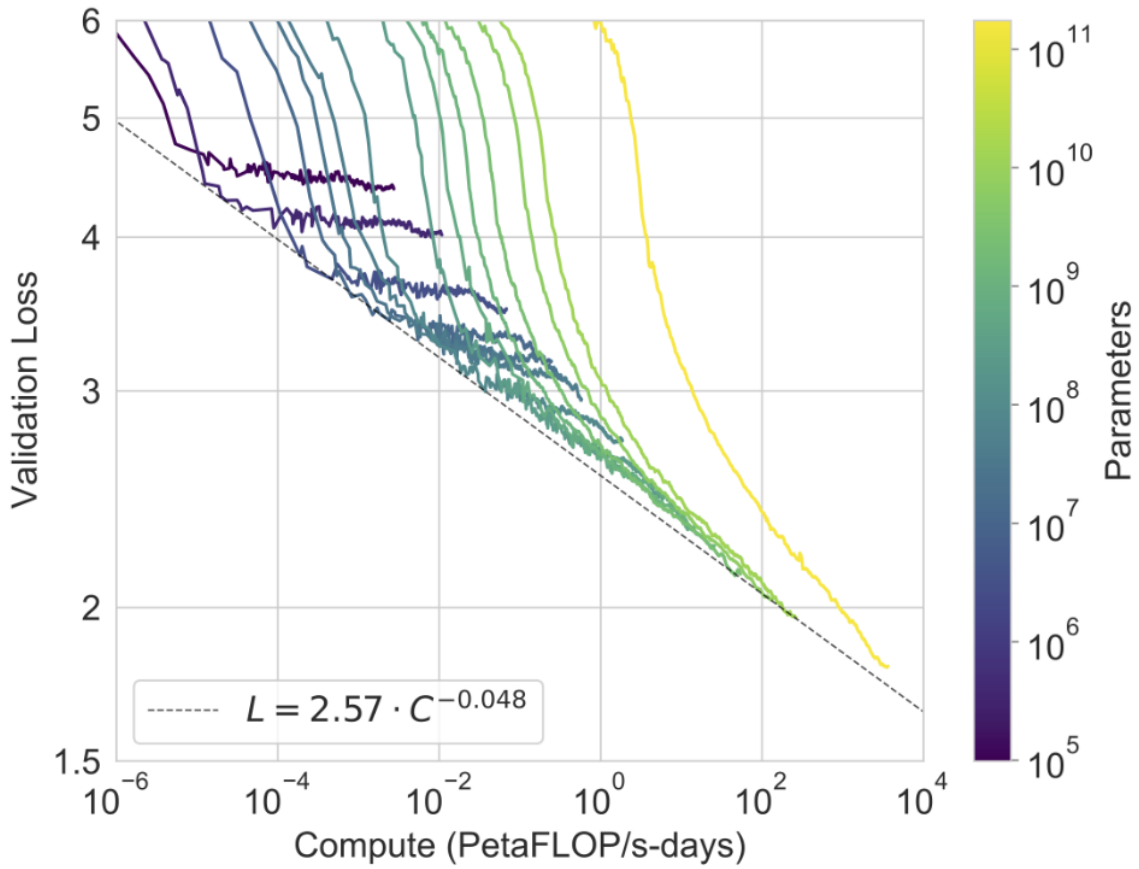
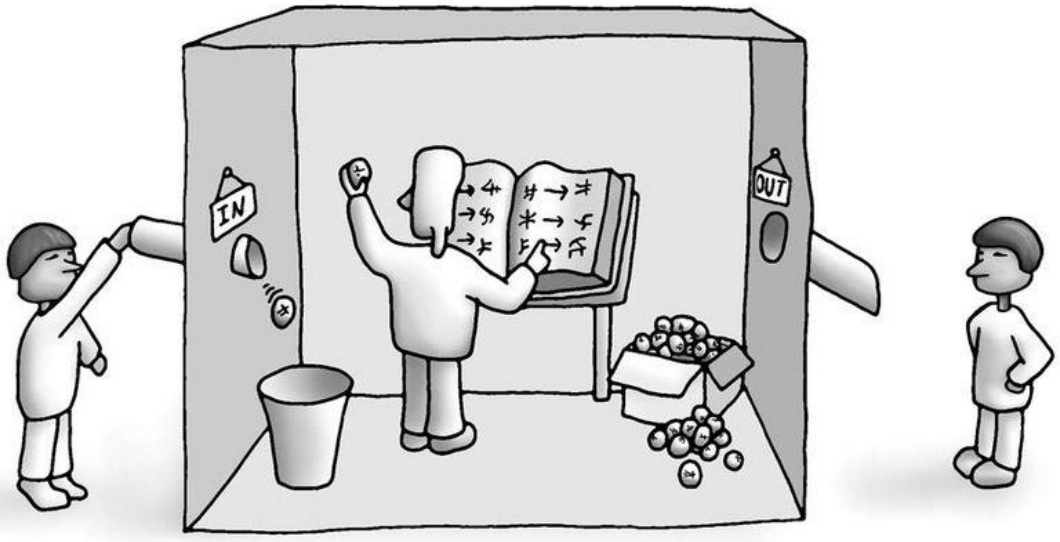


Figure 3.1: Smooth scaling of performance with compute. Performance (measured in terms of cross-entropy validation loss) follows a power-law trend with the amount of compute used for training. The power-law behavior observed in [KMH+20] continues for an additional two orders of magnitude with only small deviations from the predicted curve. For this figure, we exclude embedding parameters from compute and parameter counts.

Language Models are Few-Shot Learners - https://arxiv.org/abs/2005.14165

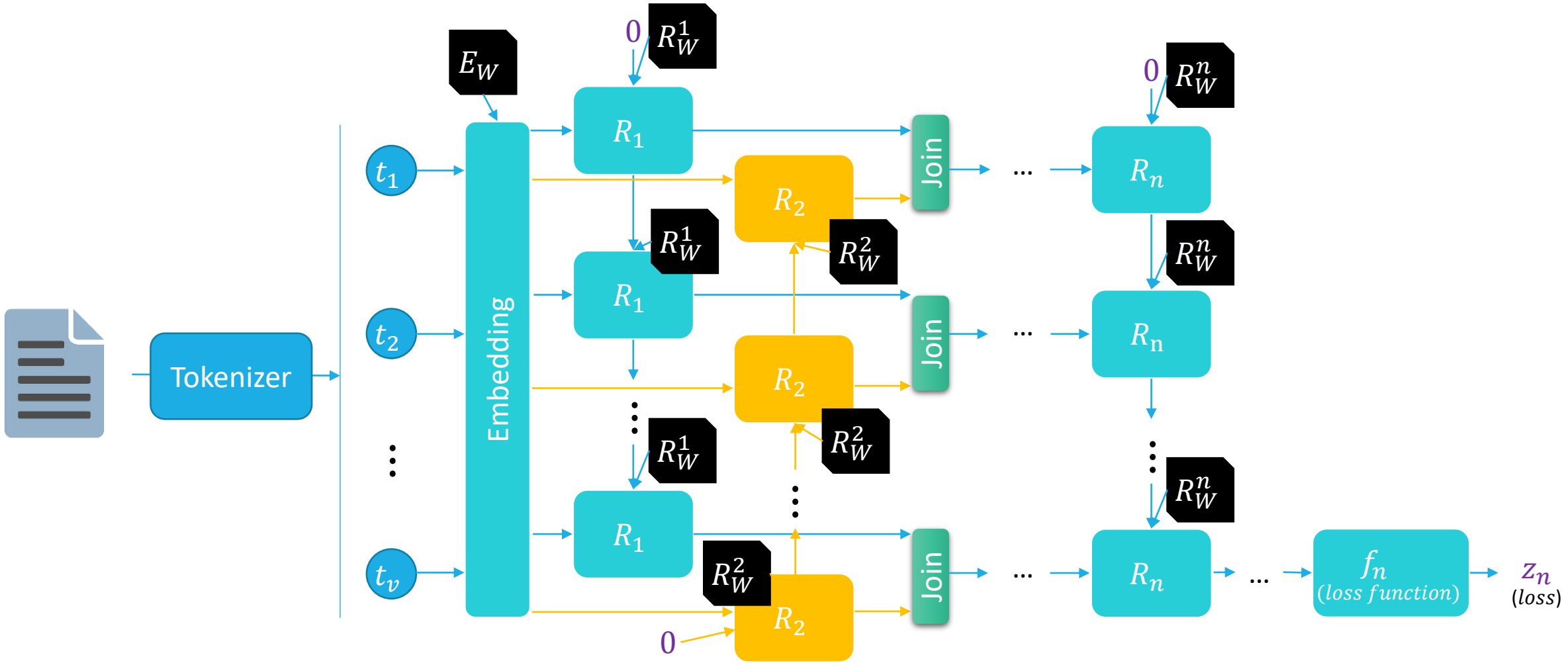
The future


Geoffrey Hinton @geoffreyhinton · 10 jun.
Extrapolating the spectacular performance of GPT3 into the future suggests that the answer to life, the universe and everything is just 4.398 trillion parameters.
63
664
3,5 mil
François Chollet @fchollet · 28 abr. 2019
Training ever bigger convnets and LSTMs on ever bigger datasets gets us closer to Strong AI -- in the same sense that building taller towers gets us closer to the **moon**.
46
651
2,4 mil


IN
OUT

Extras

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

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>

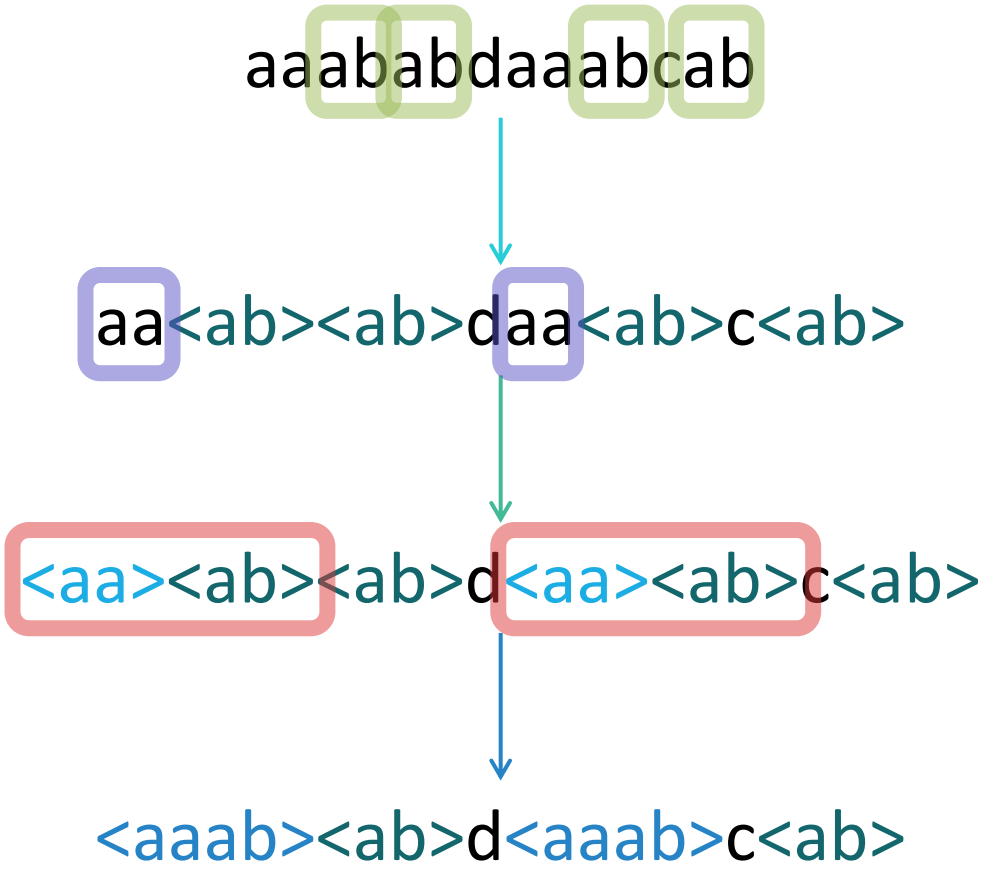
All Bedroom<END>

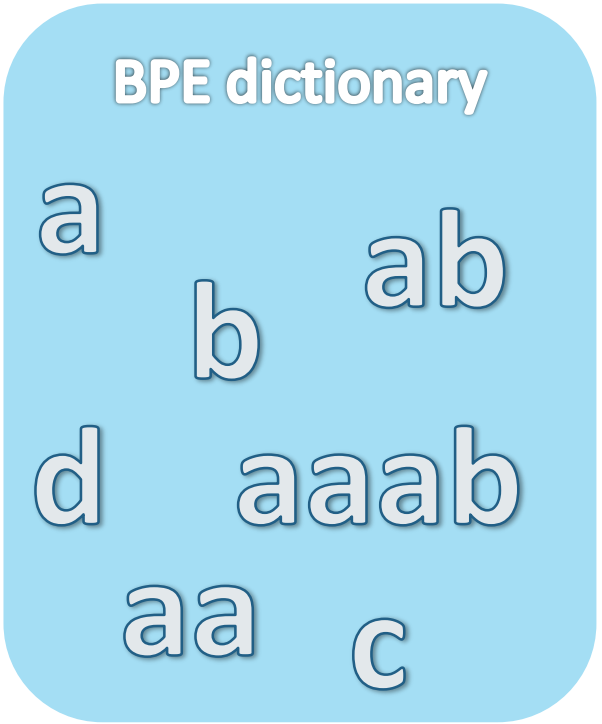
Alive a Fall<END>

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

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

Learned tokenizer: Byte Pair Encoding





Tokenizing at a sub-word level can be useful to deal with unknown words.

Initialize dictionary with all characters (or bytes!) present in the data.

Perform basic tokenization (whitespace).

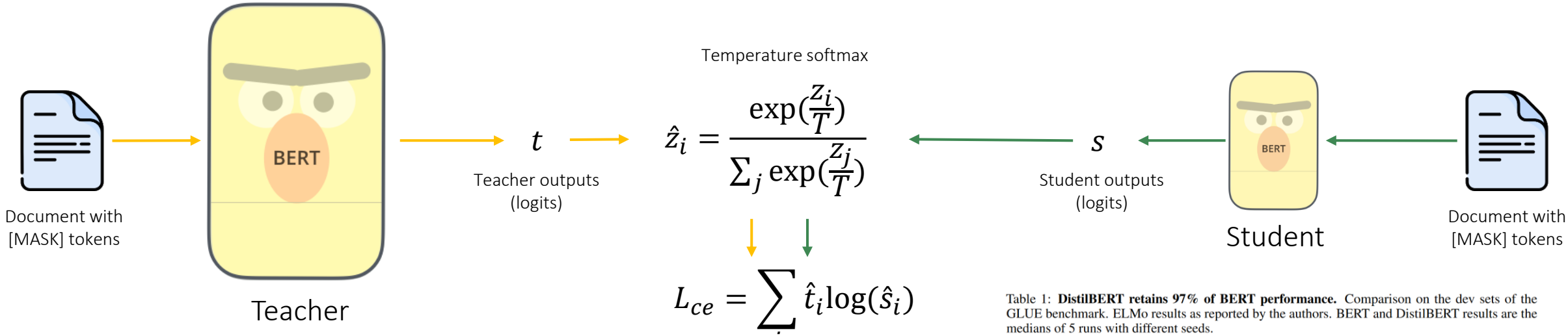
Find the couple of dictionary elements that appear jointly most frequently, and create a new entry in the dictionary with their merge. Repeat until a maximum dictionary size is reached.

https://en.wikipedia.org/wiki/Byte_pair_encoding

https://towardsdatascience.com/byte-pair-encoding-the-dark-horse-of-modern-nlp-eb36c7df4f10

DistilBERT: model distillation

After training a BERT model, it is possible to obtain a smaller, compressed version with accuracies similar to the original model



+ Distillation loss: smoother version of original labels. Intrinsically encodes BERT "dark knowledge". T hyperparameter.

+ Standard Language Model loss

+ Cosine similarity loss between teacher and student hidden vectors

Student learning loss

Table 1: **DistilBERT retains 97% of BERT performance.** Comparison on the dev sets of the GLUE benchmark. ELMo results as reported by the authors. BERT and DistilBERT results are the medians of 5 runs with different seeds.

Model	Score	CoLA	MNLI	MRPC	QNLI	QQP	RTE	SST-2	STS-B	WNLI
ELMo	68.7	44.1	68.6	76.6	71.1	86.2	53.4	91.5	70.4	56.3
BERT-base	79.5	56.3	86.7	88.6	91.8	89.6	69.3	92.7	89.0	53.5
DistilBERT	77.0	51.3	82.2	87.5	89.2	88.5	59.9	91.3	86.9	56.3

Table 2: **DistilBERT yields to comparable performance on downstream tasks.** Comparison on downstream tasks: IMDb (test accuracy) and SQuAD 1.1 (EM/F1 on dev set). D: with a second step of distillation during fine-tuning.

Model	IMDb (acc.)	SQuAD (EM/F1)
BERT-base	93.46	81.2/88.5
DistilBERT	92.82	77.7/85.8
DistilBERT (D)	-	79.1/86.9

Table 3: **DistilBERT is significantly smaller while being constantly faster.** Inference time of a full pass of GLUE task STS-B (sentiment analysis) on CPU with a batch size of 1.

Model	# param. (Millions)	Inf. time (seconds)
ELMo	180	895
BERT-base	110	668
DistilBERT	66	410

Smaller, faster, cheaper, lighter: Introducing DistilBERT, a distilled version of BERT - https://medium.com/huggingface/distilbert-8cf3380435b5

SANH et al – DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter - https://arxiv.org/pdf/1910.01108.pdf

Afi Escuela

© 2021 Afi Escuela. Todos los derechos reservados.