Deep Learning models for ASR

2. Transformer

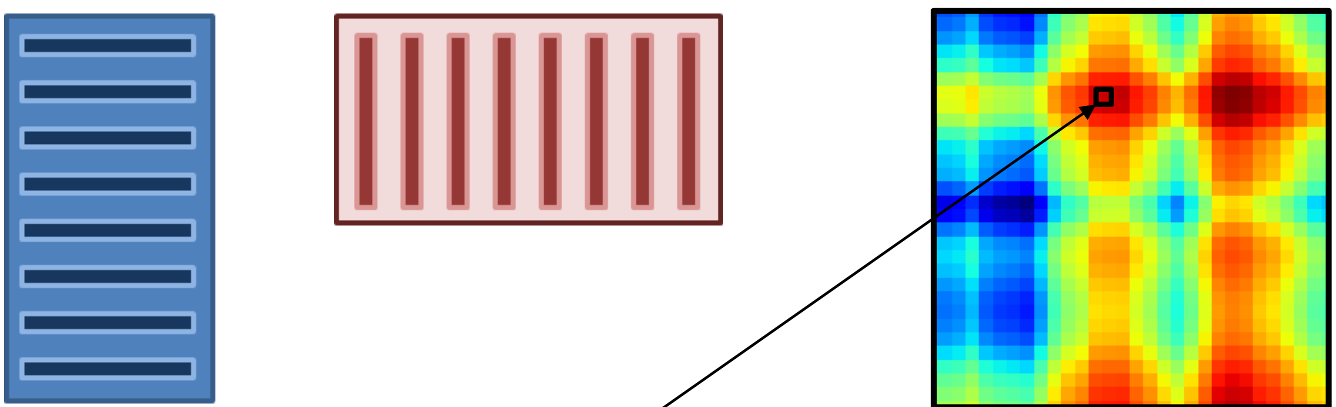
Summary

- 4 Self attention, Transformers and graph networks
  - **Introduction, self attention**
  - Transformer
  - Differentiable computers and world models

Self attention

- **Self attention**
- The core of the transformer is the self-attention operation
  - Given two sequences of vectors, k, q, of length T and dimension D
  - The product of the two matrices is used to find **similarities** between the sequences
    - In signal processing, similar to the idea of **Cross-correlation**

$$\underset{(T\times D)}{\boldsymbol{k}} \cdot \underset{(D\times T)}{\boldsymbol{q}^T} = \boldsymbol{C} \quad \text{(CrossSimilarity)}$$

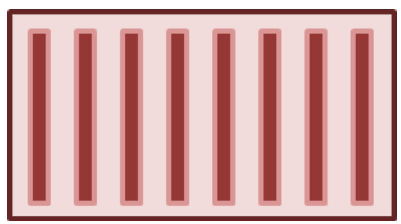


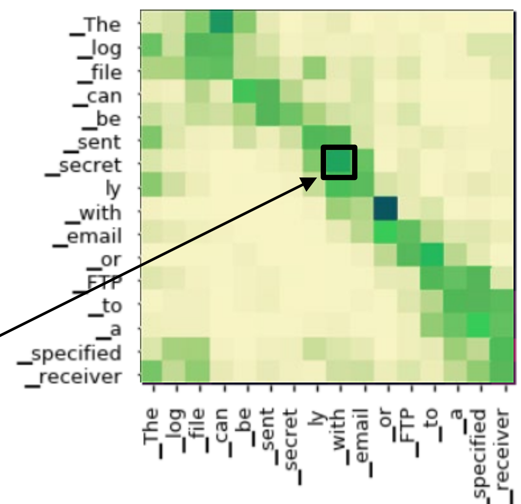
Element C_{ij}
similarity, vectors: $\boldsymbol{k}_i \cdot \boldsymbol{q}_j^T$

Self attention

- **Self attention**
- The core of the transformer is the self-attention operation
  - Given two sequences of vectors, k, q, of length T and dimension D
  - The product of the two matrices is used to find similarities between the sequences
  - A softmax operation is used to produce **mappings** of input frames to outputs (alignment matrix)

$$softmax(\underset{(T \times D)}{\boldsymbol{k}} \cdot \underset{(D \times T)}{\boldsymbol{q}^T}) = \boldsymbol{A} \quad \text{(Self attention)}$$





$$\sum_j A_{ij} = 1$$

(sum 1 cols.)

Element A_{ij}

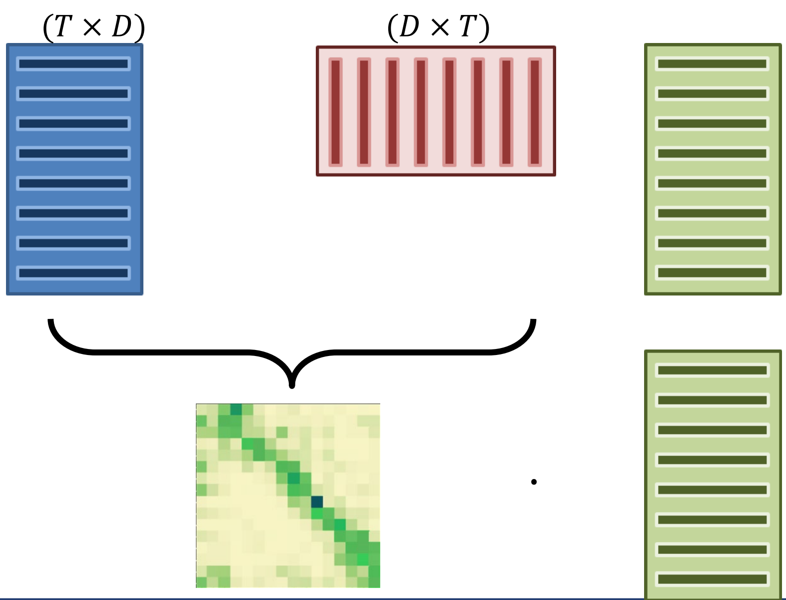
Degree of influence of input sequence item j over output i

https://nlp.seas.harvard.edu/2018/04/03/attention.html

Self attention

- **Self attention**
- The core of the transformer is the self-attention operation
  - Given two sequences of vectors, $\boldsymbol{k}, \boldsymbol{q}$, of length T and dimension D
  - The product of the two matrices is used to find similarities between the sequences
  - A softmax operation is used to produce **mappings** of input frames to outputs (alignment matrix)
  - The mapping is used to produce the output of the layer after multiplying by the values matrix
    - The process can be summarized as: the application of a dynamical mapping to the matrix $\boldsymbol{v}$

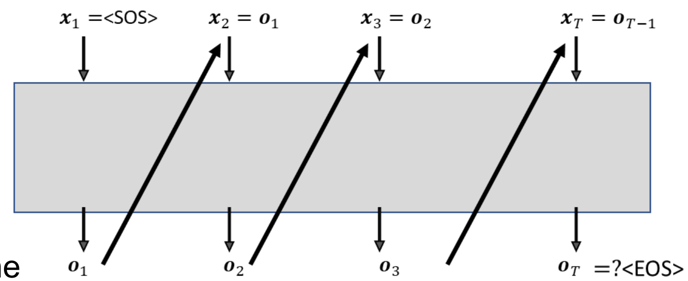
$$softmax(\boldsymbol{k} \cdot \boldsymbol{q}^T) \cdot \boldsymbol{v} = \boldsymbol{o} \quad \text{(output)}$$



- The output sequence can be **reordered**
- Each output item can be dependent of **any part of the input** sequence

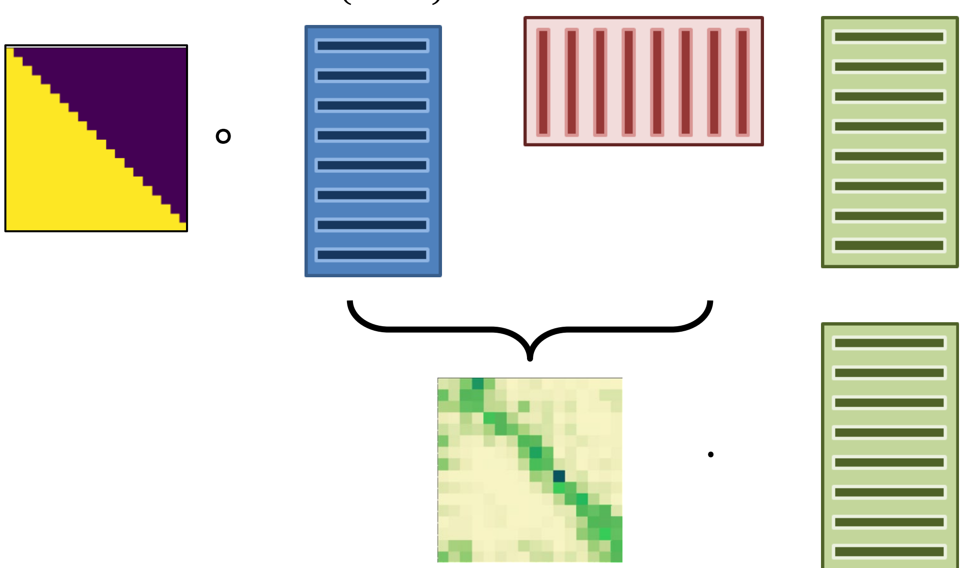
- The final output is the mapping/alignment multiplied by the sequence
- The mapping/alignment process can be interpreted as an **attention mechanism**

Self attention



- **Causal self attention**
  - To operate in generators of seq2seq architectures
    - The objective is to predict next item given previous
    - The problem if the model is not modified is that self attention allows a trivial shortcut from future token present in the intput to the Desired output
    - This is solved by multiplying the attention alignment with a mask matrix (upper diagonal values = 0)

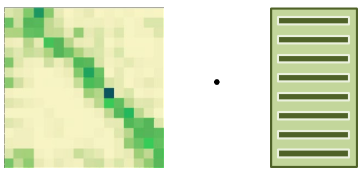
$$softmax(M \circ (\underset{(T \times D)}{\boldsymbol{k}} \cdot \underset{(D \times T)}{\boldsymbol{q}^T})) \cdot \boldsymbol{v} = \boldsymbol{o} \quad \text{(output)}$$



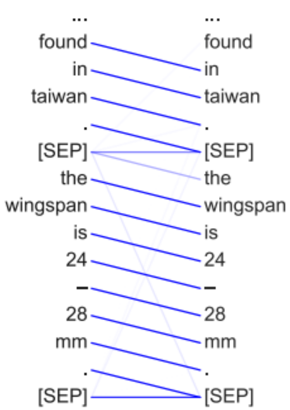
Self attention

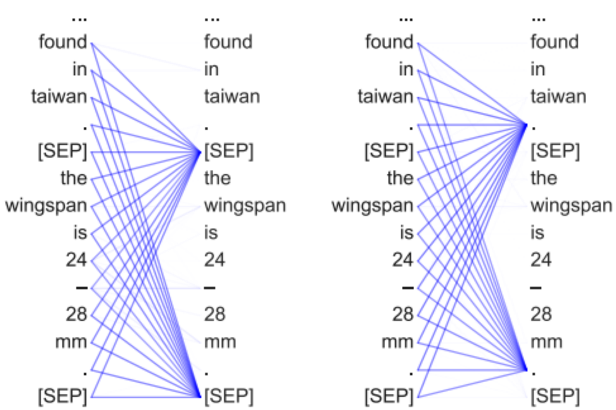
- **Self attention**
- There have been many studies trying to analyze the meaning of the mappings

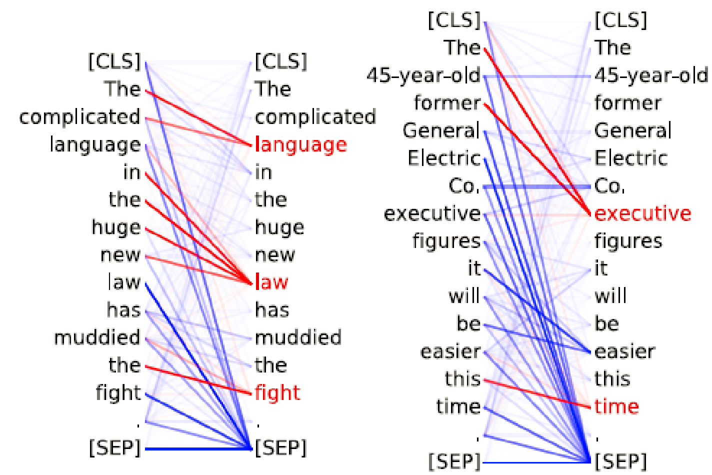
$$softmax(\boldsymbol{k}\cdot\boldsymbol{q}^T)\cdot\boldsymbol{v}$$



- One of the classic problems of NLP has been to determine automatically the relationship of a pronoun with previous text
  - e. g. ***they*** in the example on the left







Attention to previous item | Attention to end of sentence | Noun modifiers, e.g. determiners attend to their noun

https://medium.com/synapse-dev/understanding-bert-transformer-attention-isnt-all-you-need-5839ebd39

https://medium.com/dair-ai/aspects-of-language-captured-by-bert-32bc3c54016f

Summary

- 4 Self attention, Transformers and graph networks
  - Introduction, self attention
  - **Transformer**
  - Differentiable computersand world models

Transformer

- **Transformer**

  - **Attention is all you need (**Vaswani **2017)** > 60k cites

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N. Kaiser L, Polosukhin, I. (2017). Attention is all you need. Advances in neural information processing systems, 30, 5998-6008..

  - A new type of sequence models
  - The basis model for most state of the art results nowadays:
    - Translation
    - Language modeling
    - NLP: question answering, summary
    - Language understanding,
    - Speech recognition (ASR), synthesis
    - Image recognition, segmentation, detection

  - Can be used:
    - Autoregressive way: $p(x_i|x_{i-1}, x_{i-2}, \ldots)$
    - Predicting oclusions: $p(x_t|x_1, \ldots x_{t-1}, x_{t+1}, \ldots x_T)$
    - Global sequence predictions
    - Seq2seq (for translation)
    - Embedding extraction (representation learning)

Attention Is All You Need

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

Abstract

The dominant sequence transduction models are based on complex recurrent or convolutional neural networks that include an encoder and a decoder. The best performing models also connect the encoder and decoder through an attention mechanism. We propose a new simple network architecture, the Transformer, based solely on attention mechanisms, dispensing with recurrence and convolutions entirely. Experiments on two machine translation tasks show these models to be superior in quality while being more parallelizable and requiring significantly less time to train. Our model achieves 28.4 BLEU on the WMT 2014 English-to-German translation task, improving over the existing best results, including ensembles, by over 2 BLEU. On the WMT 2014 English-to-French translation task, our model establishes a new single-model state-of-the-art BLEU score of 41.8 after training for 3.5 days on eight GPUs, a small fraction of the training costs of the best models from the literature. We show that the Transformer generalizes well to other tasks by applying it successfully to English constituency parsing both with large and limited training data.

Transformer

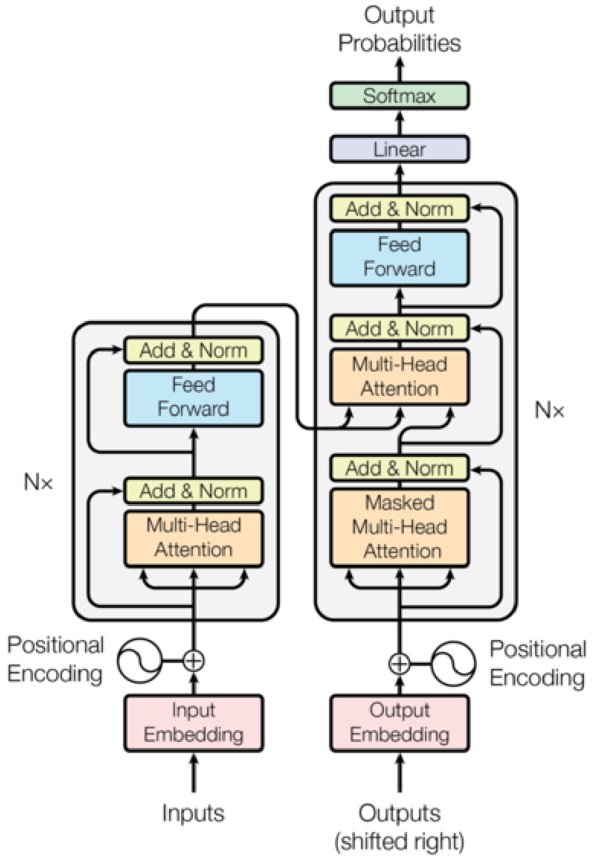
- ## Transformer

- **Attention is all you need** (Vaswani **2017)**

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N. Kaiser L, Polosukhin, I. (2017). Attention is all you need. Advances in neural information processing systems, 30, 5998-6008..

- Architecture of the system
  - The most general model has two parts:
    - **Encoder / Decoder**
    - Similar to seq2seq architecture
    - Some application only use Encoder
  - Encoder creates a processed representation for the input sentence
  - Decoder predicts next symbol/word given previously generated
  - No LSTMs or convolutions (origin of the title)
  - **Self attention** (multihead) is the main computational unit:
    - Other layers used:
      - **Embedding** layer for discrete inputs
      - **Layer normalization**
      - **Positional encoding**: to provide information about the sequence index
      - **Feed forward**: a simple MLP with a hidden layer

$$FFN(x) = \max(0, \boldsymbol{x} \cdot \boldsymbol{W}_1 + b_1) \cdot \boldsymbol{W}_2 + b_2$$



Transformer

- **Transformer**

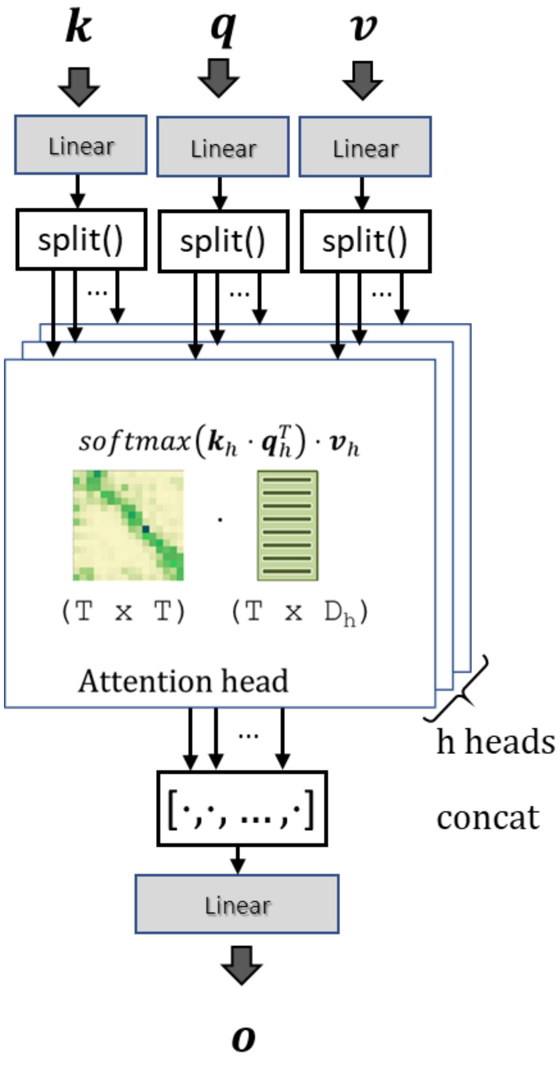
  - **Attention is all you need** (Vaswani **2017)**

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N. Kaiser L, Polosukhin, I. (2017). Attention is all you need. Advances in neural information processing systems, 30, 5998-6008..

- **Multi-Head attention**
  - The self attention is repeated in parallel **h times**, called heads (versions)
  - First a linear transformation is applied to each input sequence: $\boldsymbol{k}, \boldsymbol{q}, \boldsymbol{v}$
    - This allows each signal to represent different information,
    Since the encoder uses the same input $\boldsymbol{x}$, for $\boldsymbol{k}, \boldsymbol{q}, \boldsymbol{v}$
  - The vectors of matrices $\boldsymbol{k}, \boldsymbol{q}, \boldsymbol{v}$ are split in h submatrices: $\boldsymbol{k}_h, \boldsymbol{q}_h, \boldsymbol{v}_h$
  - The self-attention is then used with each subvector

$$Attention(\boldsymbol{q}_h, \boldsymbol{k}_h, \boldsymbol{v}_h) = softmax\left(\frac{\boldsymbol{k}_h \cdot \boldsymbol{q}_h^T}{\sqrt{D_h}}\right) \cdot \boldsymbol{v}_h$$

    - The scale normalization (dividing $\sqrt{D_h}$) improves softmax behavior (like temperature)
  - The resulting processed sequences are concatenated to obtain the same dimension as $\boldsymbol{v}$
  - Finally a linear layer is applied to obtain the output



Transformer

- ## Transformer

  - **Attention is all you need (**Vaswani **2017)**

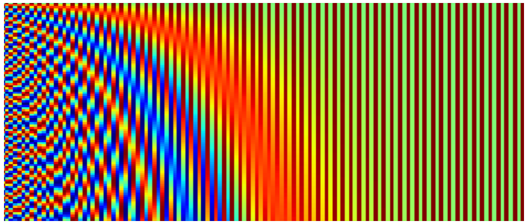
Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N. Kaiser L, Polosukhin, I. (2017). Attention is all you need. Advances in neural information processing systems, 30, 5998-6008..

- **Positional embedding**
  - The positional embedding is a matrix given to the network to allow relative distance measurements between items in the sequence
  - The expression is the following

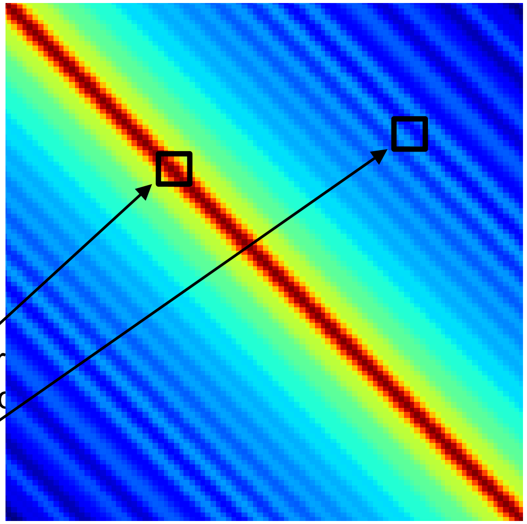
$$PE_{(pos,2i)} = sin(pos/10000^{2i/d_{\text{model}}})$$
$$PE_{(pos,2i+1)} = cos(pos/10000^{2i/d_{\text{model}}})$$

  - As illustrative example of how are distances between vectors of the matr the figure on the right illustrates the similarity product of the positional embedd vectors

Positional embedding matrix



Positional embedding similarities



- Vectors close in the sequence have high similarity
- Low similarity means a higher distance in the sequence

Transformer

- **Transformer**
  - **Layer normalization (**Ba **2016)**

Ba, J. L., Kiros, J. R., & Hinton, G. E. (2016). Layer normalization. arXiv preprint arXiv:1607.06450.

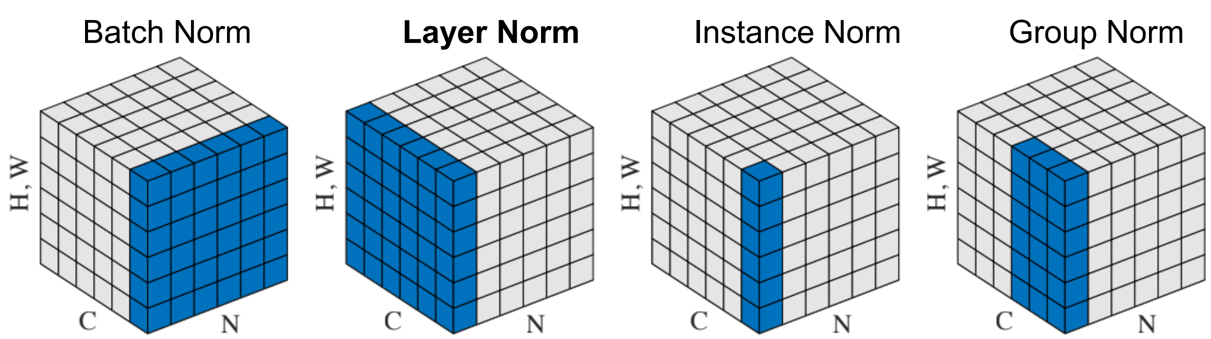
  - **Instance Normalization(**Wu **2016)**

Ulyanov, D., Vedaldi, A., & Lempitsky, V. (2016). Instance normalization: The missing ingredient for fast stylization. arXiv preprint arXiv:1607.08022.

  - **Group Normalization(**Wu **2018)**

Wu, Y., & He, K. (2018). Group normalization. In Proceedings of the European conference on computer vision (ECCV) (pp. 3-19).

- **Layer normalization**
  - The layer normalization is an alternative to the batch normalization
  - The objective and mechanism is similar to BN
  - The mean and std is calculated over the signal **channel dimensions** instead of minibatch



Transformer

Word representation

- Every Word/part of Word is represented by a **vector** (32k in LLAMA)
- We can imagine a toy representation: every word is represented

by a long vector of thousands of simple properties (sparse vector)

palabra	rey	reina	gato	árbol	ráiz	gustará	odiaba
propiedades							
persona	✓	✓					
monarquía	✓	✓					
vegetal				✓	✓		
animal			✓				
masculino	✓		✓	✓			
femenino		✓			✓		
positivo						✓	
negativo							✓
verbo						✓	✓
presente							
pasado							✓
futuro						✓	

- Similar words -> **similar representations**

A change in meaning can be associated with changing one of those aspects; they resemble more a Chinese ideogram than a representation through a phonetic alphabet.

木 → 本

Árbol Raíz

Transformer

Word representation

- The analogy is a strong simplification but we can extract two important conclusions:

palabra	rey	reina	gato	árbol	ráiz	gustará	odiaba
propiedades							
persona	✓	✓					
monarquía	✓	✓					
vegetal				✓	✓		
animal			✓				
masculino	✓		✓	✓			
femenino		✓			✓		
positivo						✓	
negativo							✓
verbo						✓	✓
presente							
pasado							✓
futuro						✓	

- Word comparing mechanism:

  Two words are **similar** if they share many of the properties (scalar producto of vectors)

- Word transformations:

  We can **transform** a word to approximate to another word by the manipulation of its properties

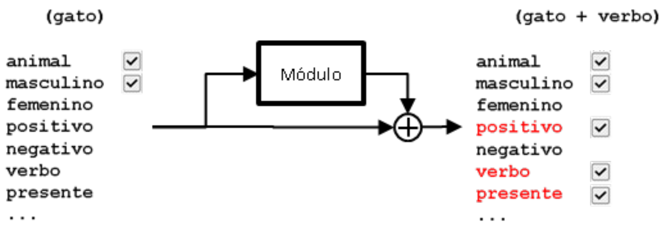
```
rey - hombre + mujer -> (is close to) reina
```

- Other analogies obtained from Spanish word2vec:

```
tenis – raqueta + bate              -> béisbol
motocicleta – motor + pedales       -> bicicleta
amar – positivo + negativo          -> odiar
parís – francia + españa            -> madrid
```

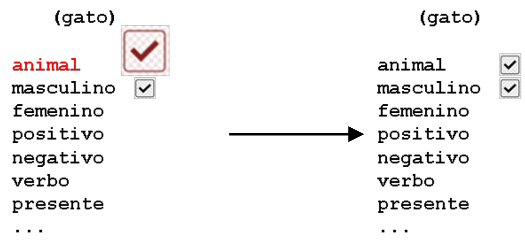
Transformer

▪ Transformers (summary)

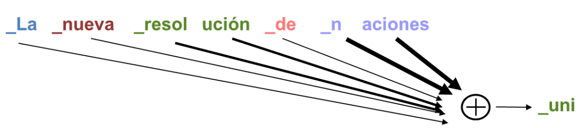


Information is processed **incrementally** (residual)

Thought vectors (G. Hinton)



It is important for convergence to **normalice** vectors, better before blocks



Two types of modules
Independent vector processing
- FFN

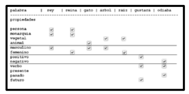
Mixing vectors
- **attention mechanism** (self) (weighted sum)

Transformer

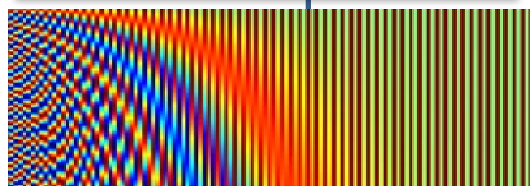
GPT-3 13B / LLAMA13B

_La	_nueva	_resol	ución	_de	_n	aciones

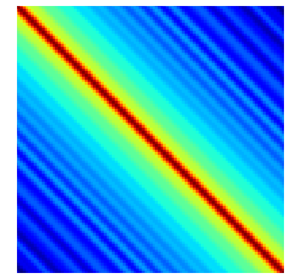
(decoder)



1.2 Adding **positional embedding**

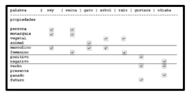


(relative distance)



Transformer

_La	_nueva	_resol	ución	_de	_n	aciones

(word embeddings) 

la, última, resolución, de, naciones

w_1 w_2 w_3 w_4 w_5

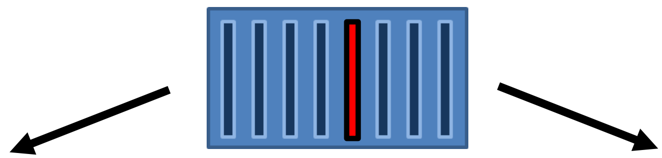
$w_2 = 1125$

dictionary

…
1124 ultílogo
1125 última
1126 ultimación
…

GPT-3 13B / LLAMA13B

Dictionary size 32k tokens



Embedding Layer

Index column of embeddig matrix

$$\boldsymbol{x}_2 = \boldsymbol{W}_e[:, w_2]$$

Linear Layer (not efficient)

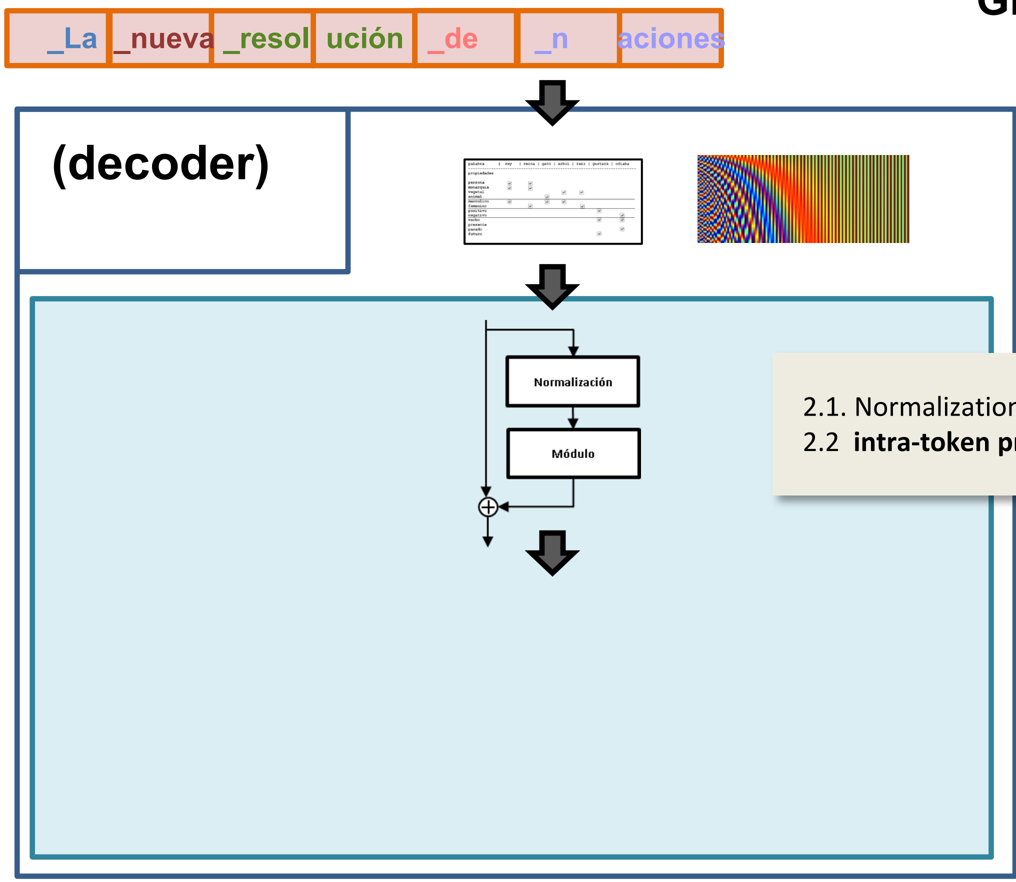
$$\boldsymbol{w}_2 = one\_hot(w_2)$$

$$\boldsymbol{w}_2 = [0, 0, \ldots, \underset{\uparrow index:\ 1125}{1}, \ldots, 0, 0$$

$$\boldsymbol{x}_2 = \boldsymbol{w}_2 \cdot \boldsymbol{W}_e$$

Transformer

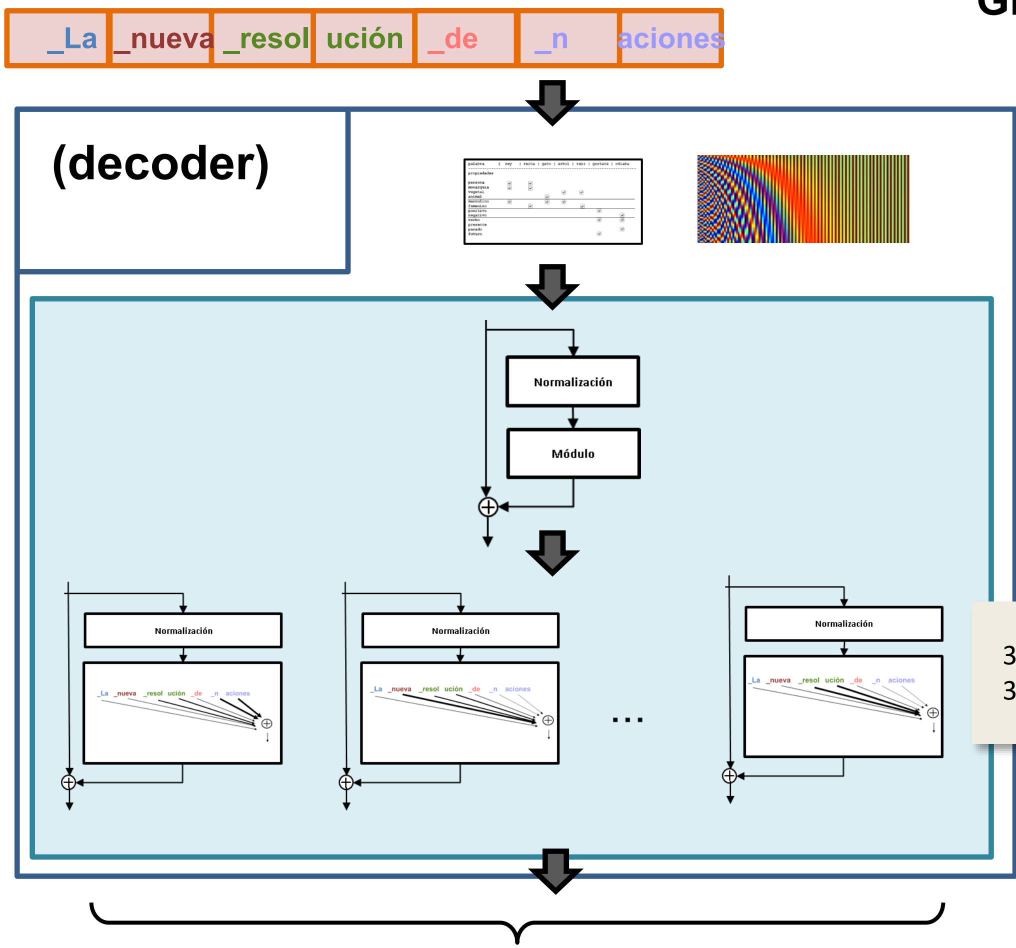
GPT-3 13B / LLAMA13B



2.1. Normalization
2.2 **intra-token processing**

Transformer

GPT-3 13B / LLAMA13B

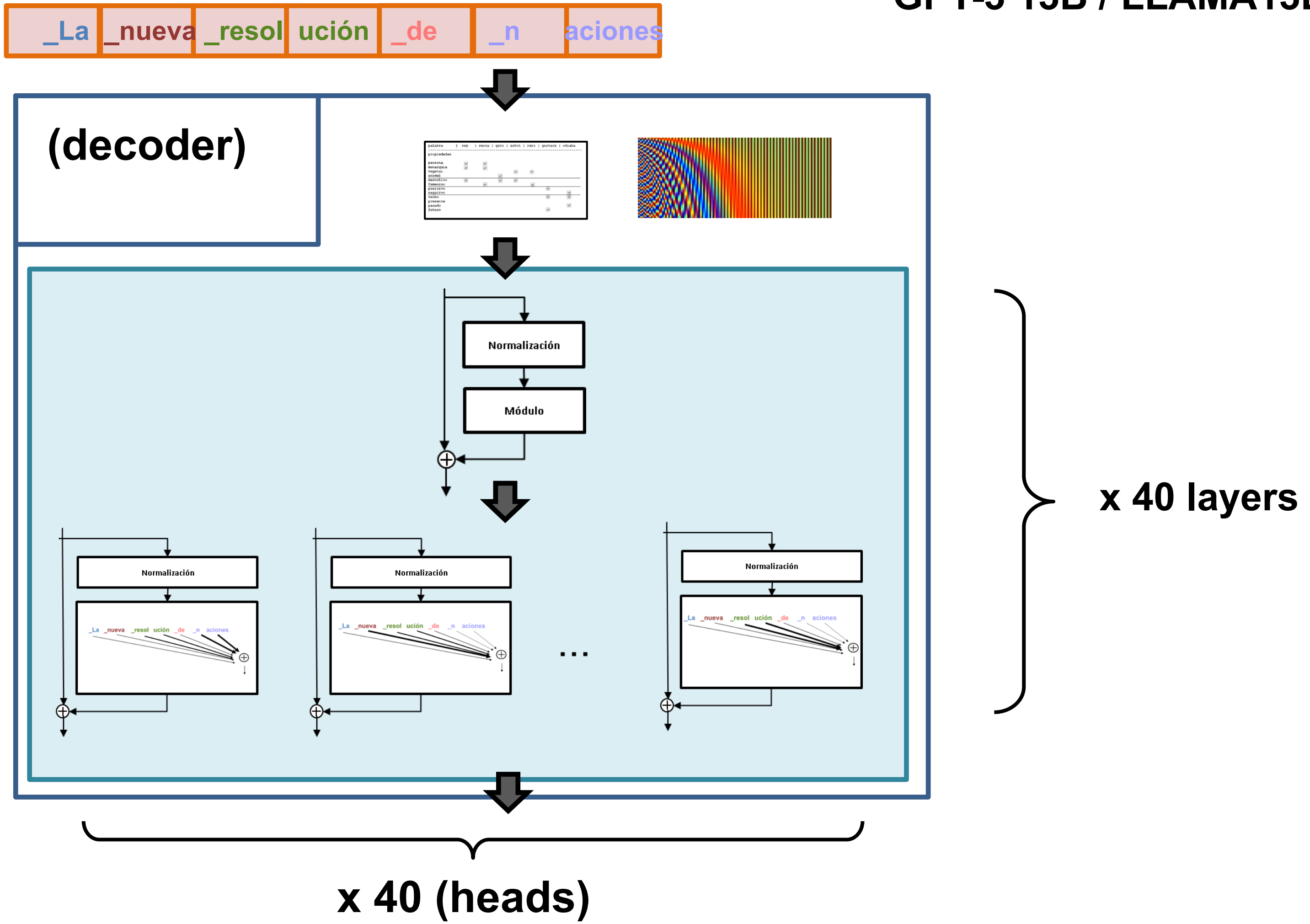


x 40 (heads / cabezales)

3.1 Normalization Self Attention
3.2 **inter token processing**

Transformer

GPT-3 13B / LLAMA13B



Summary

- 4 Self attention, Transformers and graph networks
  - Introduction, self attention
  - **Transformer**
    - **Examples**
  - Differentiable computers and world models

Transformer

▪ Transformer models

- **ELMO (**Peters **2018)**

Peters, M. E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., & Zettlemoyer, L. (2018). Deep contextualized word representations. arXiv preprint arXiv:1802.05365...

- Trained as lm model, next word given previous,
  - Also bidirectional, (reverse order)
  - It concatenates both outputs

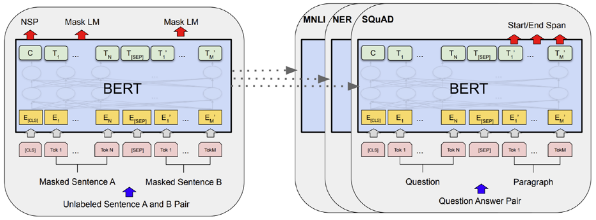
$$p(x_1,\ldots,x_n)=\prod_{i=1}^{n}p(x_i \mid x_1,\ldots,x_{i-1})$$

$$p(x_1,\ldots,x_n)=\prod_{i=1}^{n}p(x_i \mid x_{i+1},\ldots,x_n)$$ (reverse order)

- **BERT (**Devlin **2018)**

Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805..

- BERT: Bidirectional Encoder Representations from Transformers
- Once it is trained the representation is used for task specific application (idea of representation learning, unsupervised learning)
  - For example text classification, question answering
- **Task 1**
- The transformer is used to predict masked items
- The masks are generated as follows:
  - 80% were replaced by the **‘<MASK>’** token
    - Example: “My dog is **<MASK>**”
  - 10% were replaced by a random token
    - Example: “My dog is apple”
  - 10% were left intact
    - Example: “My dog is hairy”
- **Task 2**
- The second task: given two sentences the transformer has to predict if one follows the other



Pre-training Fine-Tuning

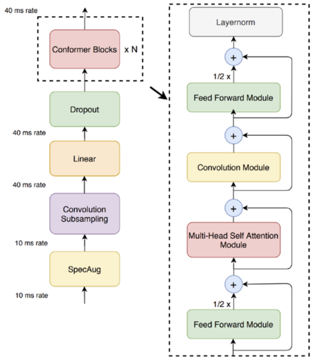
BERT BASE (L=12, H=768, A=12, Total Parameters=**110M**)
BERT LARGE (L=24, H=1024, A=16, Total Parameters=**340M**)

Transformer

- ## Transformer models
- **Conformer(**Gulati **2020)**

Gulati, A., Qin, J., Chiu, C. C., Parmar, N., Zhang, Y., Yu, J., ... & Pang, R. (2020). C onformer: Convolution-augmented Transformer for Speech Recognition. arXiv preprint arXiv:2005.08100.

- combine CNNs and transformers
  - model both **local** and **global** dependencies
  - audio sequences, ASR

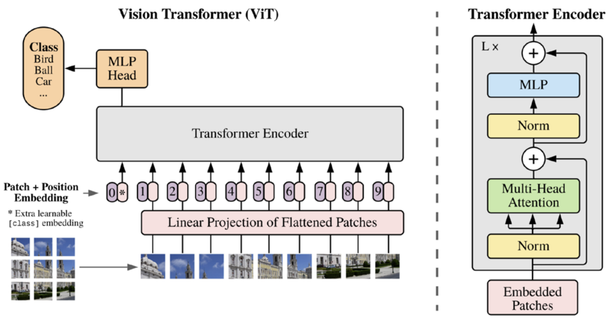


Method	#Params (M)	WER Without LM		WER With LM	
		testclean	testother	testclean	testother
Hybrid					
Transformer [33]	-	-	-	2.26	4.85
CTC					
QuartzNet [9]	19	3.90	11.28	2.69	7.25
LAS					
Transformer [34]	270	2.89	6.98	2.33	5.17
Transformer [19]	-	2.2	5.6	2.6	5.7
LSTM	360	2.6	6.0	2.2	5.2
Transducer					
Transformer [7]	139	2.4	5.6	2.0	4.6
ContextNet(S) [10]	10.8	2.9	7.0	2.3	5.5
ContextNet(M) [10]	31.4	2.4	5.4	**2.0**	4.5
ContextNet(L) [10]	112.7	**2.1**	4.6	**1.9**	4.1
Conformer (Ours)					
Conformer(S)	10.3	**2.7**	**6.3**	**2.1**	**5.0**
Conformer(M)	30.7	**2.3**	**5.0**	**2.0**	**4.3**
Conformer(L)	118.8	**2.1**	**4.3**	**1.9**	**3.9**

- **Vision Transformer(**Dosovitskiy **2020)**

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., ... & Uszkoreit, J. (2020). An image is worth 16x16 words: Transformers for image recognition at scale. arXiv preprint arXiv:2010.11929...

- State of the art results in image tasks have been obtained using transformer to predict pixel values
- Images are decomposed in 16x16 patches read left-to-right
- Imagenet 88.55%   (for example ResNExt 101: 85.4%) Top1 accuracy



Model	Layers	Hidden size D	MLP size	Heads	Params
ViT-Base	12	768	3072	12	86M
ViT-Large	24	1024	4096	16	307M
ViT-Huge	32	1280	5120	16	632M

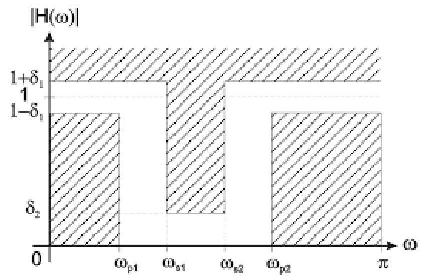
Model size and parameters

Summary

- 4 Self attention, Transformers and graph networks
  - Introduction, self attention
  - Transformer
  - **Differentiable computers and world models**

Differentiable computers

- Engineering problem:



- Specifications and constraints to meet
- We choose a method/algorithm to solve them
- Finally, we check the solution and sometimes we can rank them

- In Machine Learning the approach is different:

```
entrada      | salida
------------------------------
0, 2, 1, 5   | 0, 1, 2, 5
4, 5, 3      | 3, 4, 5
0, 1         | 0, 1
8, 4, 5      | 4, 5, 8
...
```

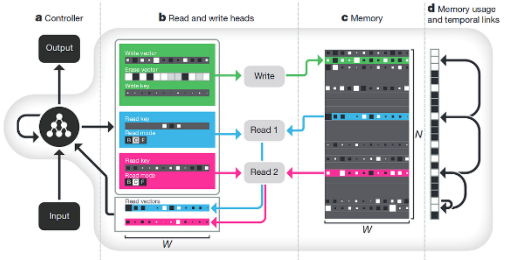
- We select simples inputs, desired outputs

- We train a model by fitting some desired loss function

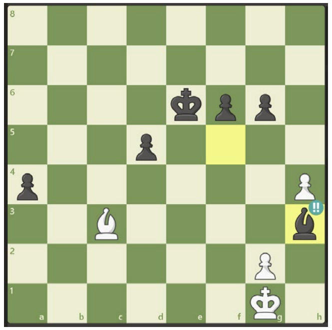
- When assesing the quality we can have errors even for small trivial problems

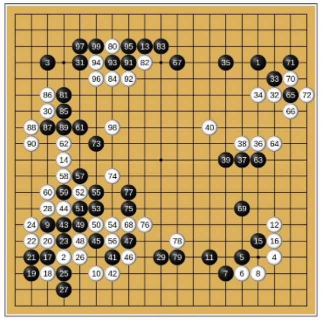
Differentiable computers

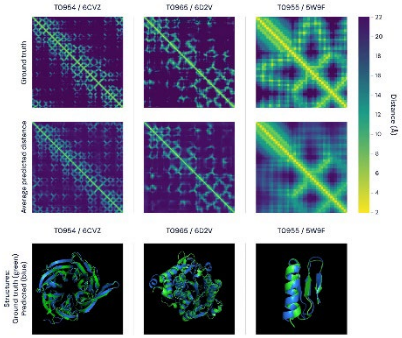
- **Differentiable computer**
  - 2014, Neural Turing Machines
  - 2016, Differentiable Neural Computer
  - 2017, Transformers



- **Is it useful ? Solving problems for unknown algorithms**
  - This capability stands out **when an algorithm is not known** to solve the task: translating between languages, summarizing texts, evaluating a strategic position in a complex game, predicting protein folding, and so on.



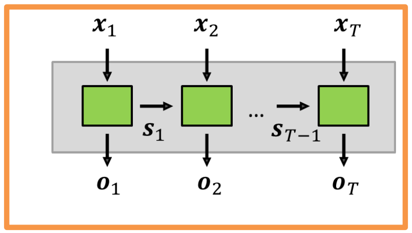

First 99 moves



https://www.deepmind.com/blog/alphafold-using-ai-for-scientific-discovery-2020

Differentiable computers

- **Applications:** NTM, DNCs
- Memory with two heads: writting, Reading (size N)



Controller (LSTM)

$\boldsymbol{k}_t$ $(key/address)$

$$\boldsymbol{r}_t = \sum_i w_t(i) \cdot \boldsymbol{M}_t[i,:] \qquad \sum_i w_t(i) = 1$$

Read head

$$\boldsymbol{M}_t[i,:] = \boldsymbol{M}_{t-1}[i,:](1 - w_t(i)\boldsymbol{e}_t) + w_t(i)\boldsymbol{a}_t$$

Write head:
erasing $\boldsymbol{e}_t$ adding $\boldsymbol{a}_t$

Memory $\boldsymbol{M}_t$ $(N \times D)$

- The memory indexing is done by measuring similarity: cosine distance
- We compute the similarity of the “index” from the controller:

called **key, $\boldsymbol{k}_t$,** to all the memorry positions

- Weigths are normalized to sum 1 using **softmax**

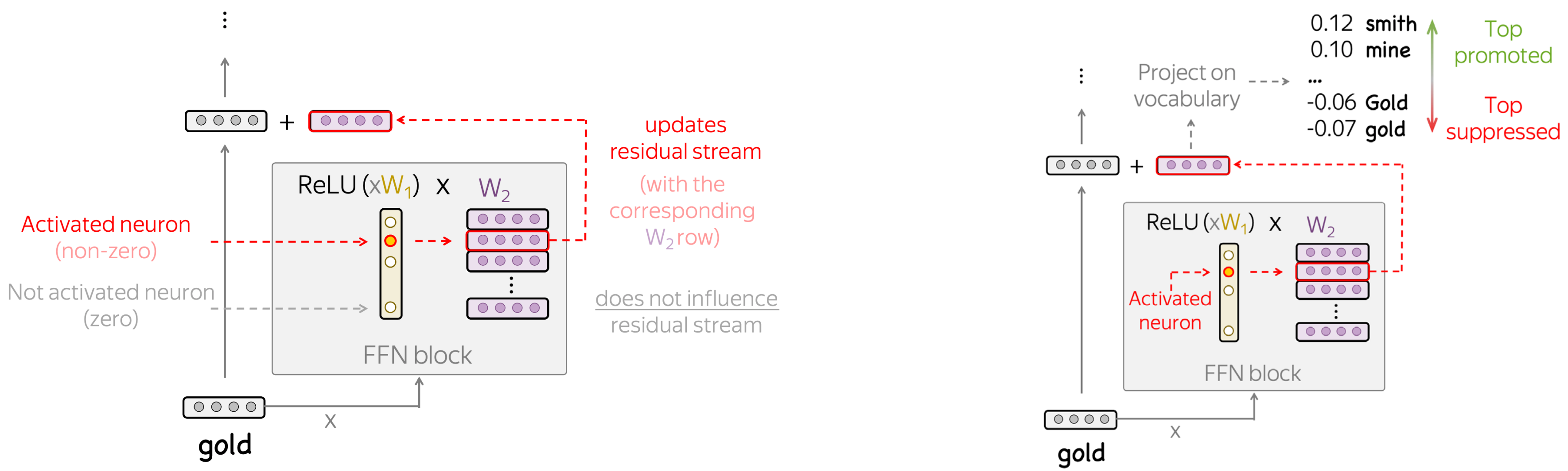
$$K[\mathbf{u},\mathbf{v}] = \frac{\mathbf{u}\cdot\mathbf{v}}{||\mathbf{u}||\cdot||\mathbf{v}||}$$

$$w_t(i) = \frac{exp(\beta_t K[\boldsymbol{k}_t, \boldsymbol{M}_t[i,:]])}{\sum_j exp(\beta_t K[\boldsymbol{k}_t, \boldsymbol{M}_t[j,:]])}$$

  - This can be interpreted as a posterior probability: the higher most similar to key, and it Will be selected to read/write

Transformer

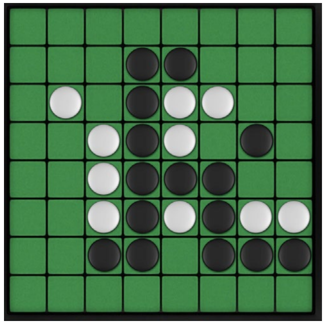
- **FFN interpretation as memory layer**

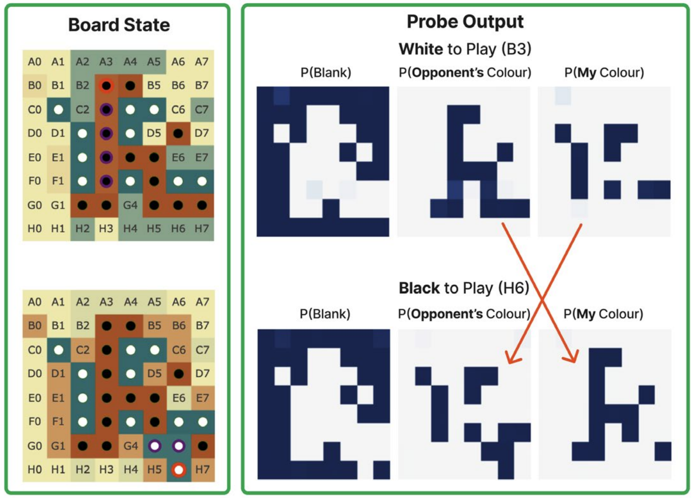


https://lena-voita.github.io/posts/neurons_in_llms_dead_ngram_positional.html

World models

- **Transformer: world models**
  - Model trained to predict next move in reversi (otelo).
    - Image left. Board after moves: F4, F3, D2, F5, G2, F2, G3, C4, E5, F6, D6, E2, B4, C5, G7, C1, G6, F7, G5, C3, B3, **H6**





  - The model has no visual aid as input but it could be shown that the internal representation had encoded the position of both players. (image right)

https://www.lesswrong.com/posts/nmxzr2zsjNtjaHh7x/actually-othello-gpt-has-a-linear-emergent-world

World models

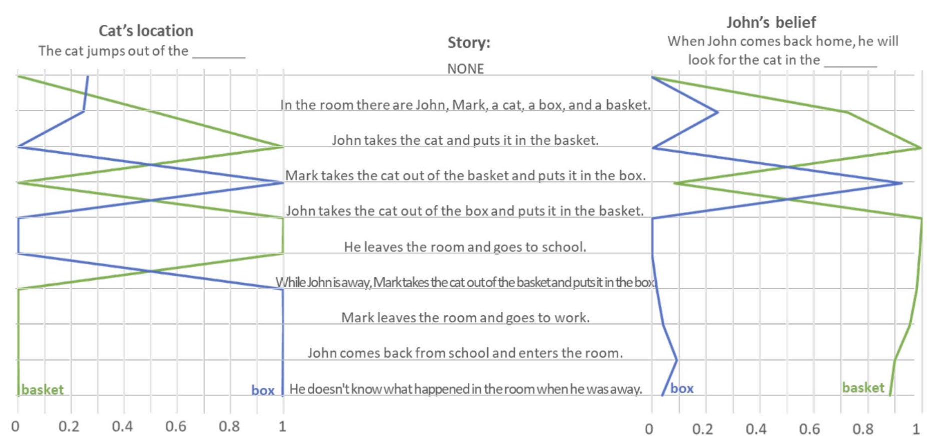
- **Transformer: world models**
  - There are numerous recent works in which LMs are evaluated or trained to solve a multitude of tasks and their ability to model scenarios based on previous context.

Task 1: Single Supporting Fact
Mary went to the bathroom.
John moved to the hallway.
Mary travelled to the office.
Where is Mary? A:office

Task 7: Time Reasoning
In the afternoon Julie went to the park.
Yesterday Julie was at school.
Julie went to the cinema this evening.
Where did Julie go after the park? A:cinema
Where was Julie before the park? A:school

Task 8: Positional Reasoning
The triangle is to the right of the blue square.
The red square is on top of the blue square.
The red sphere is to the right of the blue square.
Is the red sphere to the right of the blue square? A:yes
Is the red square to the left of the triangle? A:yes

Xiang, J et al (2023). Language Models Meet World Models: Embodied Experiences Enhance Language Models.



*Theory of Mind May Have Spontaneously Emerged in Large Language Models Authors: Michal Kosinski*1*

VIVOLAB Universidad Zaragoza