# Recent Language Models

**Sanghyun Seo**

**Oct 18, 2019**

**Department of Computer Engineering at Dongguk University**

**Artificial Intelligence Laboratory**

# Papers

- NPLM (2003)
  - Bengio, Yoshua, et al. "A neural probabilistic language model." *Journal of machine learning research* 3.Feb (2003): 1137-1155.
- Word2vec (2013)
  - Mikolov, Tomas, et al. "Distributed representations of words and phrases and their compositionality." *Advances in neural information processing systems*. 2013.
- RNNLM (2010)
  - Mikolov, Tomáš, et al. "Recurrent neural network based language model." *Eleventh annual conference of the international speech communication association*. 2010.
- ELMo (2018)
  - Peters, Matthew E., et al. "Deep contextualized word representations." *arXiv preprint arXiv:1802.05365* (2018).
- Transformer (2017)
  - Vaswani, Ashish, et al. "Attention is all you need." *Advances in neural information processing systems*. 2017.

# Language Model

- Language model
  - The **language model** is modeling the probability of generating natural language sentences or documents
  - We can use the language model to estimate how natural a sentence or a document is, Also, with the language model, we can generate new sentences or documents
- Probabilistic Language Models
  - Language models assign a probability to each sentence

$$P(W) = P(w_0, w_1, \dots, w_n) = \prod_{i=0}^{|W|} P(w_i) = \prod_{i=1}^{|W|+1} P(w_i | w_0, \dots, w_{i-1})$$

  - Example
    - “this is a language model”

$$P(this, is, a, language, model)$$
$$= P(this)P(is|this)\ P(a|this, is)P(language|this, is, a)\ P(model|this, is, a, language)$$

# Language Model

- Probabilistic Language Models
  - Machine translation
    - $P(this\ is\ language\ model) > P(this\ are\ language\ model)$
  - Spell correction
    - $P(I\ have\ a\ runny\ nose\ ) > P(I\ have\ a\ runny\ noise)$
  - Speech recognition
    - $P(I\ need\ a\ cup\ of\ water\ ) > P(I\ neat\ a\ cup\ of\ water)$
  - Etc…

- Count based word probability
  - $P(model|this, is, a, language) = \frac{count(this\ is\ a\ language\ model)}{count(this\ is\ a\ language)}$
    - This sentence is likely. However, it is likely to be a low probability depending on the characteristics of the corpus we have
    - If we don't have "$this\ is\ a\ language$" sentence?

    → The denominator is 0
  - It is sparsity problem

# Language Model

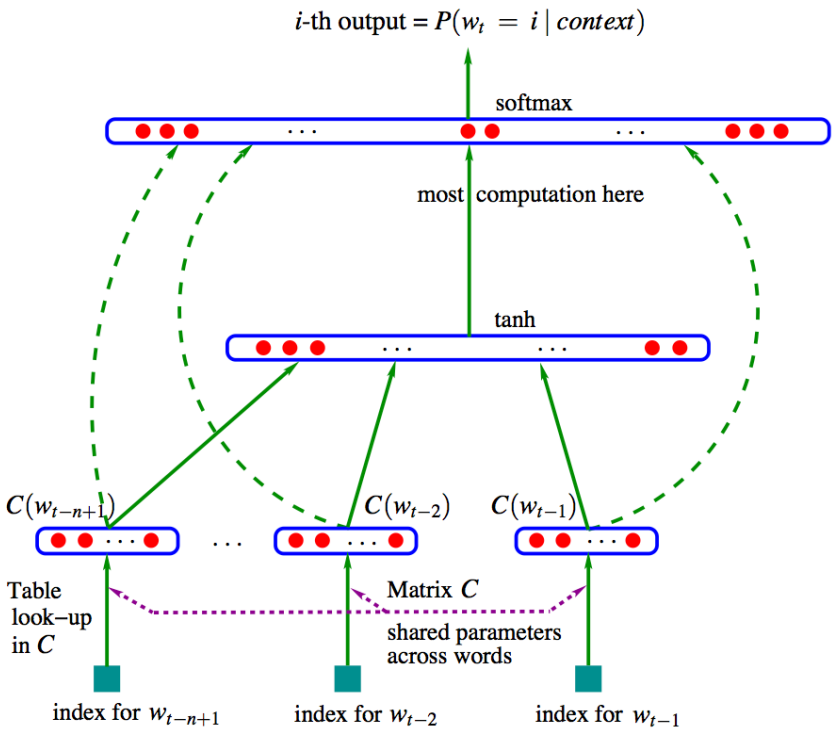
- N-gram language model
  - Unigram, Bigram, Trigram, N-gram
- Bigram example
  - $P(w_i|w_0, \dots, w_{i-1}) \approx P(w_i|w_{i-1})$
  - Example
    - "this is a language model"

$$P(this\ is\ a\ language\ model)$$
$$= P(this)P(is|this)\ P(is|this, is)(language|this, is, a)\ P(model|this, is, a, language)$$
$$\approx P(this)P(is|this)P(\text{a}|\text{is})P(language|is)P(model|language)$$

  - Still, It has a sparsity problem

# Neural Language Model

- NPLM(Neural Probabilistic Language Model)
  - $P(w_i|w_0, \ldots, w_{i-1})$ is obtained from neural networks

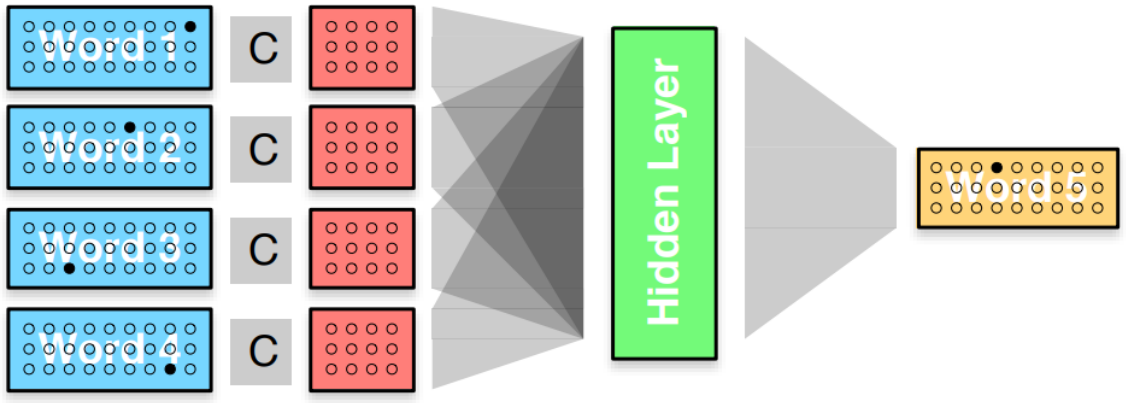



- Bengio, Yoshua, et al. "A neural probabilistic language model." *Journal of machine learning research* 3.Feb (2003): 1137-1155.

# Neural Language Model

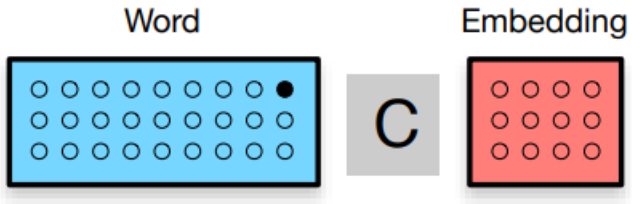
- NPLM(Neural Probabilistic Language Model)
  - Words are represented with a one-hot vector
    - This = [1, 0, 0, 0, 0 …, 0], Is = [0, 1, 0, 0, 0 …, 0], a = [0, 0, 1, 0, 0 …, 0]

    Language = [0, 0, 0, 1, 0 …, 0], Model = [0, 0, 0, 0, 1,…, 0]
  - That's a large vector → limit to 20,000 most frequent words
  - Map each word first into a lower-dimensional real-valued space



# Neural Language Model

- NPLM(Neural Probabilistic Language Model)
  - Add direct connections from embedding layer to output layer
  - Activation functions
    - input→embedding: none
    - embedding→hidden: tanh
    - hidden→output: softmax




  - By-product: embedding of word into continuous space
  - Similar contexts → similar embedding
  - Recall: distributional semantics

# Neural Language Model

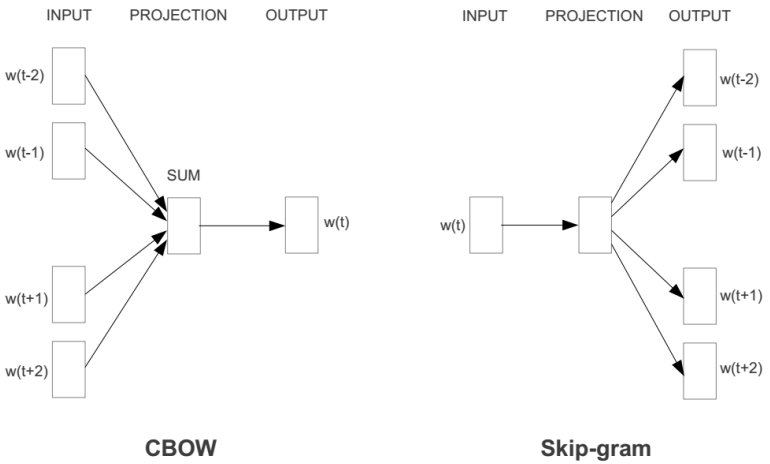
- NPLM(Neural Probabilistic Language Model)
  - Word embedding (distributed representation)

# Word Embedding

- Word embedding model
  - Sparse representation
    - one-hot encoding = [0, 0,…1,…, 0]
  - Distributed representation
    - dense vector =  = [0.2, 0.7,…1.23,…,-2.3]
  - One-hot representation → word embedding model → distributed representation(embedding vector)
  - Embedding vector
    - Low dimensional
    - Trainable
    - Float type

# Word Embedding

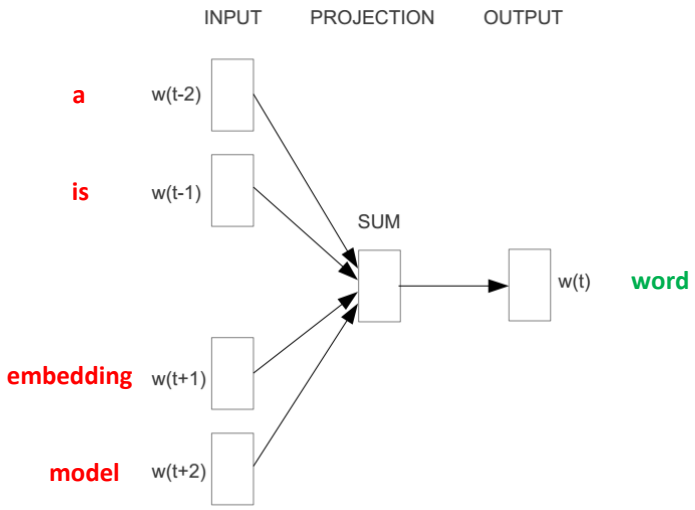
- Word2vec
  - 2 basic neural network models:
    - Continuous Bag of Word (CBOW): use a window of word to predict the middle word
    - Skip-gram (SG): use a word to predict the surrounding ones in window




- Mikolov, Tomas, et al. "Distributed representations of words and phrases and their compositionality." Advances in neural information processing systems. 2013.

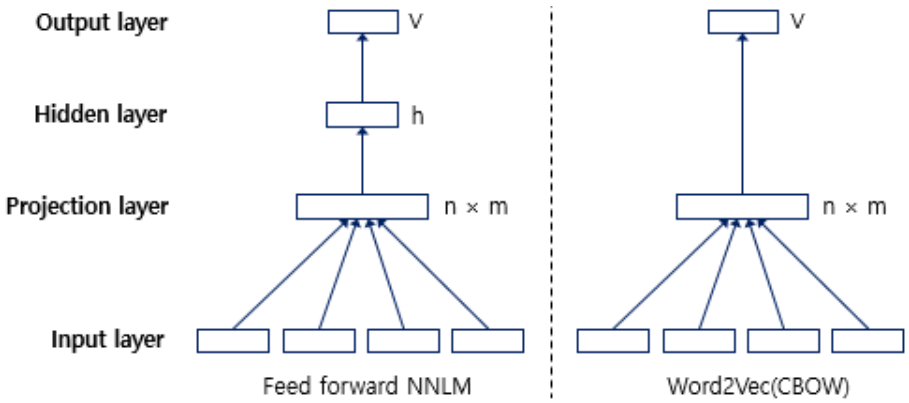
# Word Embedding

- Word2vec
  - E.g. “this is a word embedding model”
  - Window size = 2
  - $\boldsymbol{P(word|a, is, embedding, model;\ \theta)}$



# Word Embedding

- Word2vec
  - Computation
    - NPLM: (n*m)+(n*m*h)+(h*V)
    - Word2vec: (n*m)+(m*V)
    - Word2vec: (n*m)+(m*log V) with negative sampling




  - Similar language model
    - Glove, Fast-text, …

- https://wikidocs.net/22660

# Contextual Language Model

- RNNLM: recurrent neural network based language model
  - $P(w_i|w_0, \dots, w_{i-1})$ is obtained from recurrent neural networks

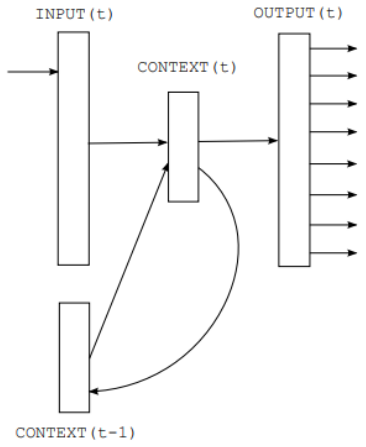
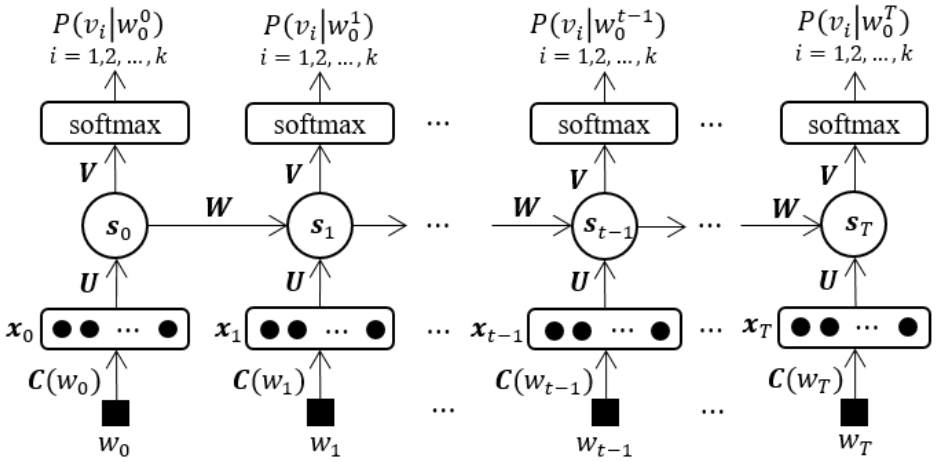



Figure 1: *Simple recurrent neural network.*




- Mikolov, Tomáš, et al. "Recurrent neural network based language model." Eleventh annual conference of the international speech communication association. 2010.
- https://dengliangshi.github.io/2017/01/01/neural-network-language-models.html

# Contextual Language Model

- ELMo: Embeddings from Language Models
  - Bidirectional language model
    - Forward pass

$$P(w_0, w_1, \dots, w_n) = \prod_{i=1}^{|W|+1} P(w_i | w_0, \dots, w_{i-1})$$

    - Backward pass

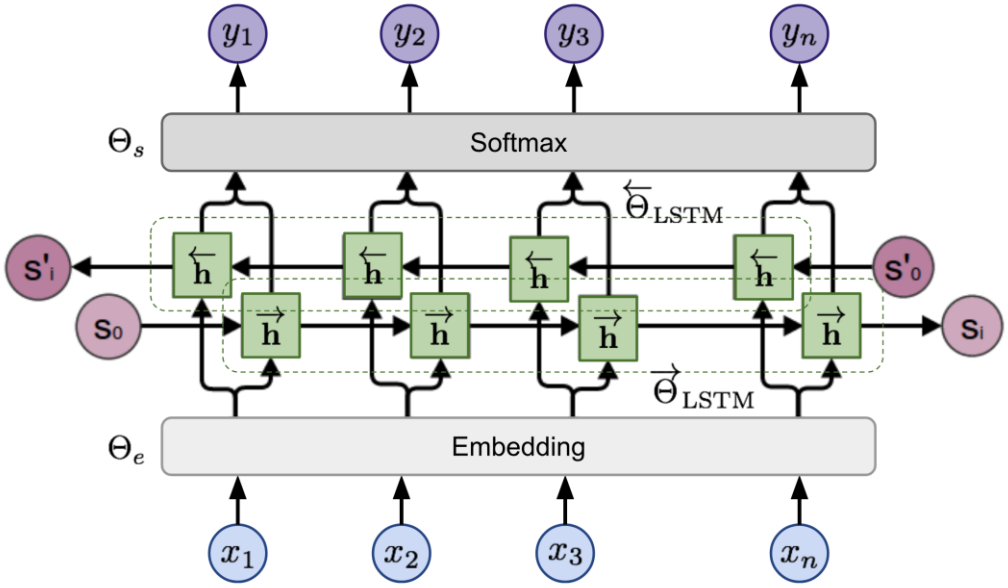
$$P(w_0, w_1, \dots, w_n) = \prod_{i=1}^{|W|+1} P(w_i | w_{i+1}, \dots, w_n)$$

- Peters, Matthew E., et al. "Deep contextualized word representations." arXiv preprint arXiv:1802.05365 (2018).

# Contextual Language Model

- ELMo: Embeddings from Language Models
  - Bidirectional language model
    - Objective function → negative log likelihood

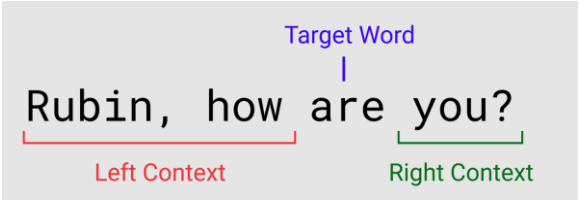
$$\mathcal{L} = -\sum_{i=1}^{n} \Big( \log p(x_i \mid x_1, \ldots, x_{i-1}; \Theta_e, \overrightarrow{\Theta}_{\text{LSTM}}, \Theta_s) + \log p(x_i \mid x_{i+1}, \ldots, x_n; \Theta_e, \overleftarrow{\Theta}_{\text{LSTM}}, \Theta_s) \Big)$$

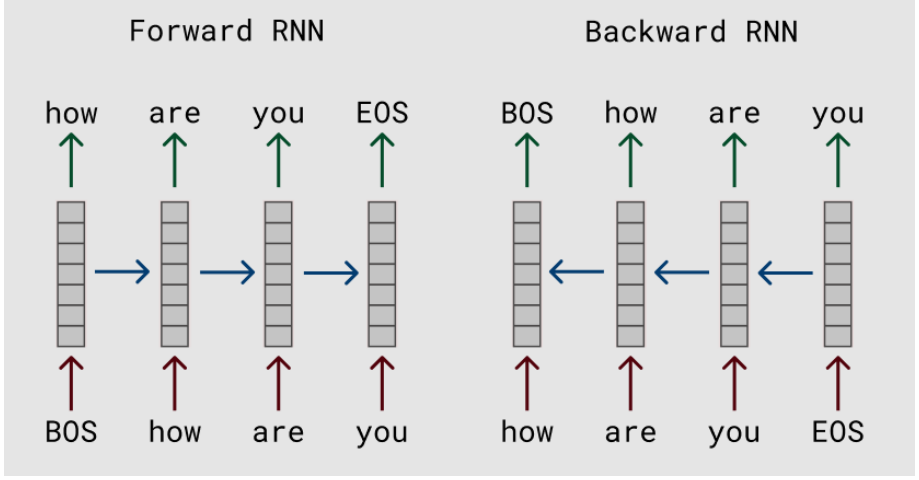


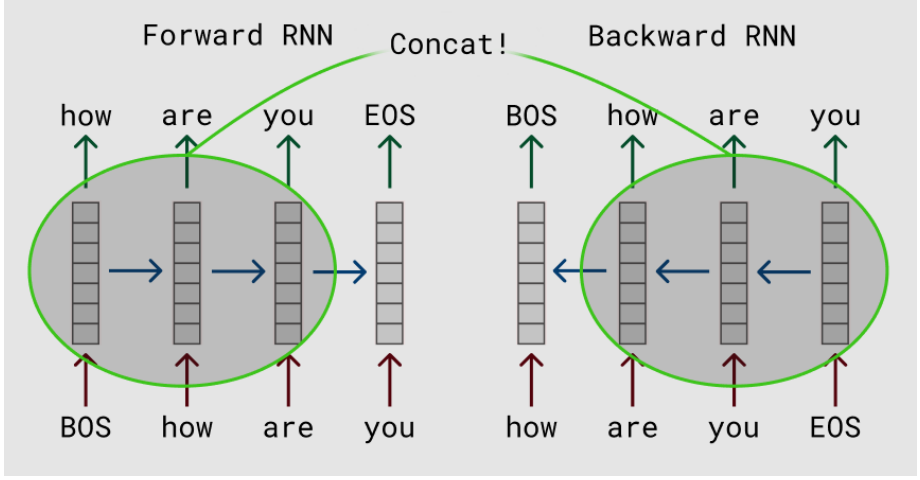

- https://www.topbots.com/generalized-language-models-cove-elmo/

# Contextual Language Model

- ELMo: Embeddings from Language Models
  - Bidirectional language model



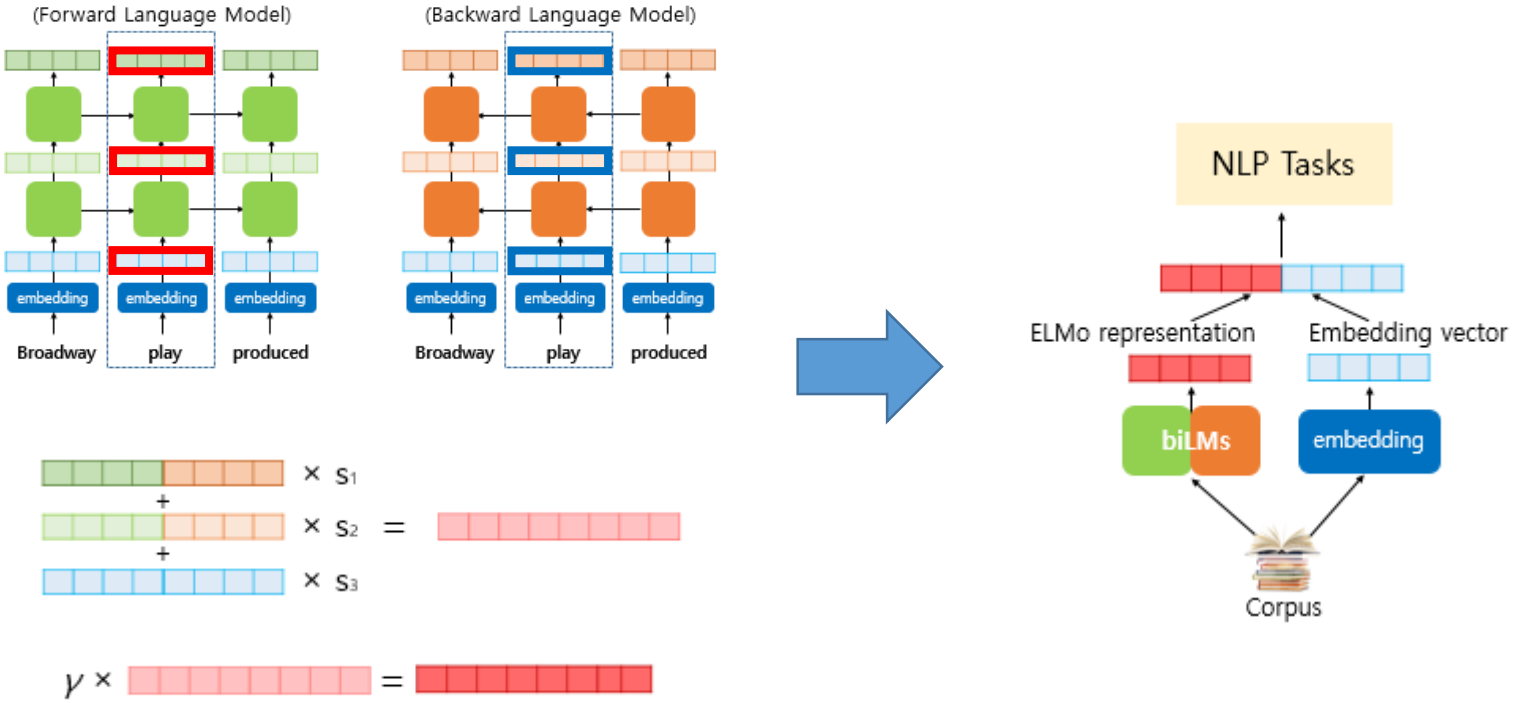




https://medium.com/@plusepsilon/the-bidirectional-language-model-1f3961d1fb27

# Contextual Language Model

- ELMo: Embeddings from Language Models
  - ELMo Representations → dynamic representations
    - Example: “Broadway play produced by him”

$$v_i = f(R_i; \Theta^{\text{task}}) = \gamma^{\text{task}} \sum_{\ell=0}^{L} s_i^{\text{task}} \mathbf{h}_{i,\ell}$$




- https://wikidocs.net/33930

# Contextual Language Model

- ELMo: Embeddings from Language Models
  - dynamic representations
  - Disambiguate both the part of the speech and word sense in the source sentence

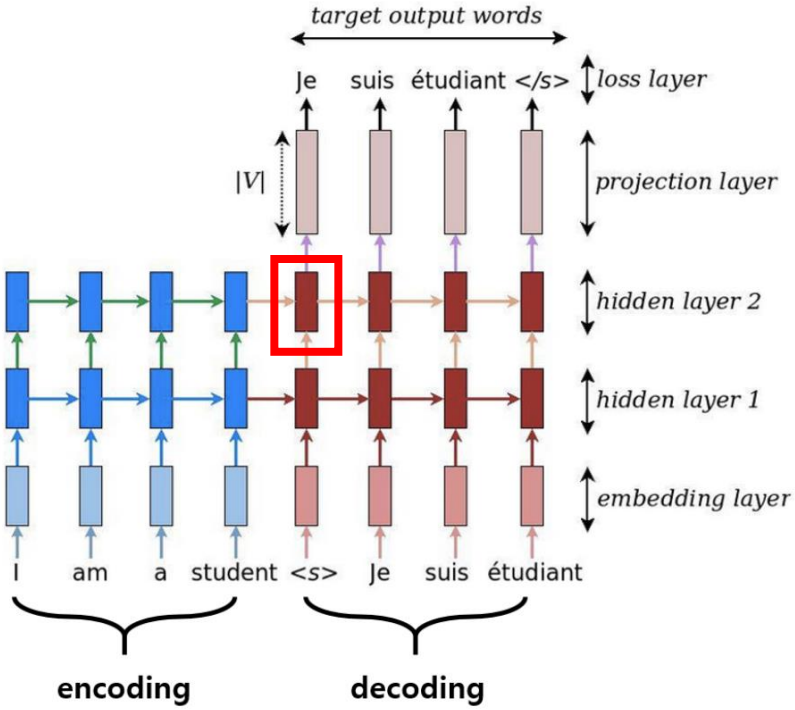
| | Source | Nearest Neighbors |
|---|---|---|
| GloVe | play | playing, game, games, played, players, plays, player, Play, football, multiplayer |
| biLM | Chico Ruiz made a spectacular play on Alusik 's grounder {...} | Kieffer , the only junior in the group , was commended for his ability to hit in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

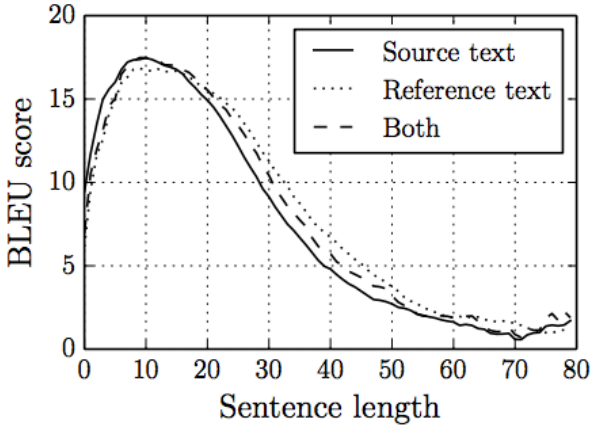
Table 4: Nearest neighbors to "play" using GloVe and the context embeddings from a biLM.


- Peters, Matthew E., et al. "Deep contextualized word representations." arXiv preprint arXiv:1802.05365 (2018).

# Transformer

- Seq2Seq (encoder-decoder)
  - Missing long-term memory
  - Vanishing gradients

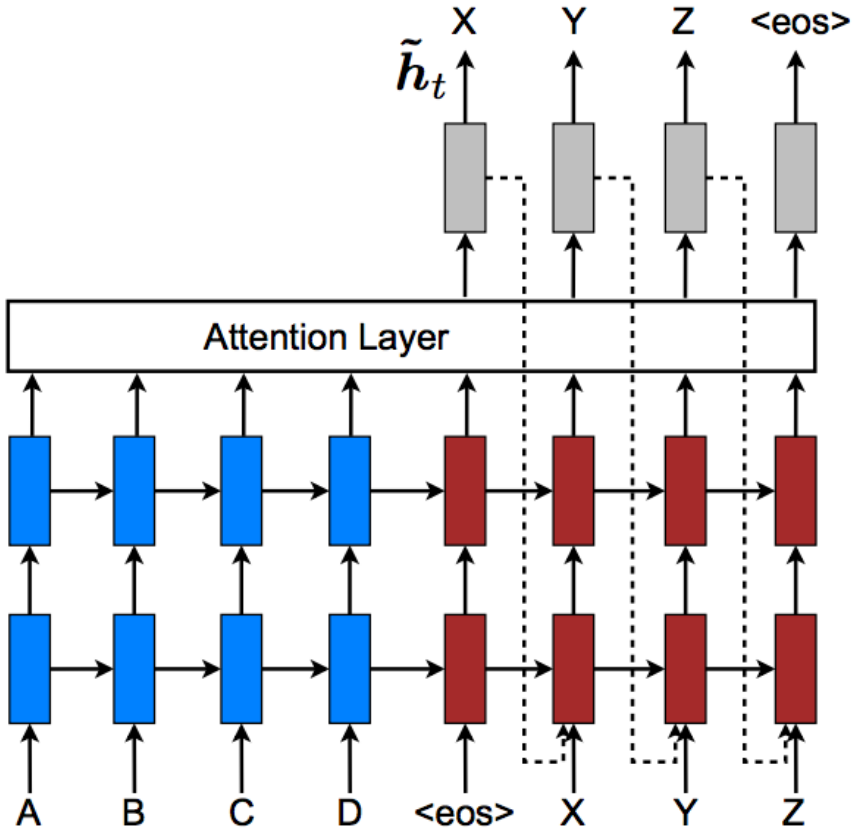



- https://towardsdatascience.com/seq2seq-model-in-tensorflow-ec0c557e560f
- Cho, Kyunghyun, et al. "On the properties of neural machine translation: Encoder-decoder approaches." arXiv preprint arXiv:1409.1259 (2014).

# Transformer

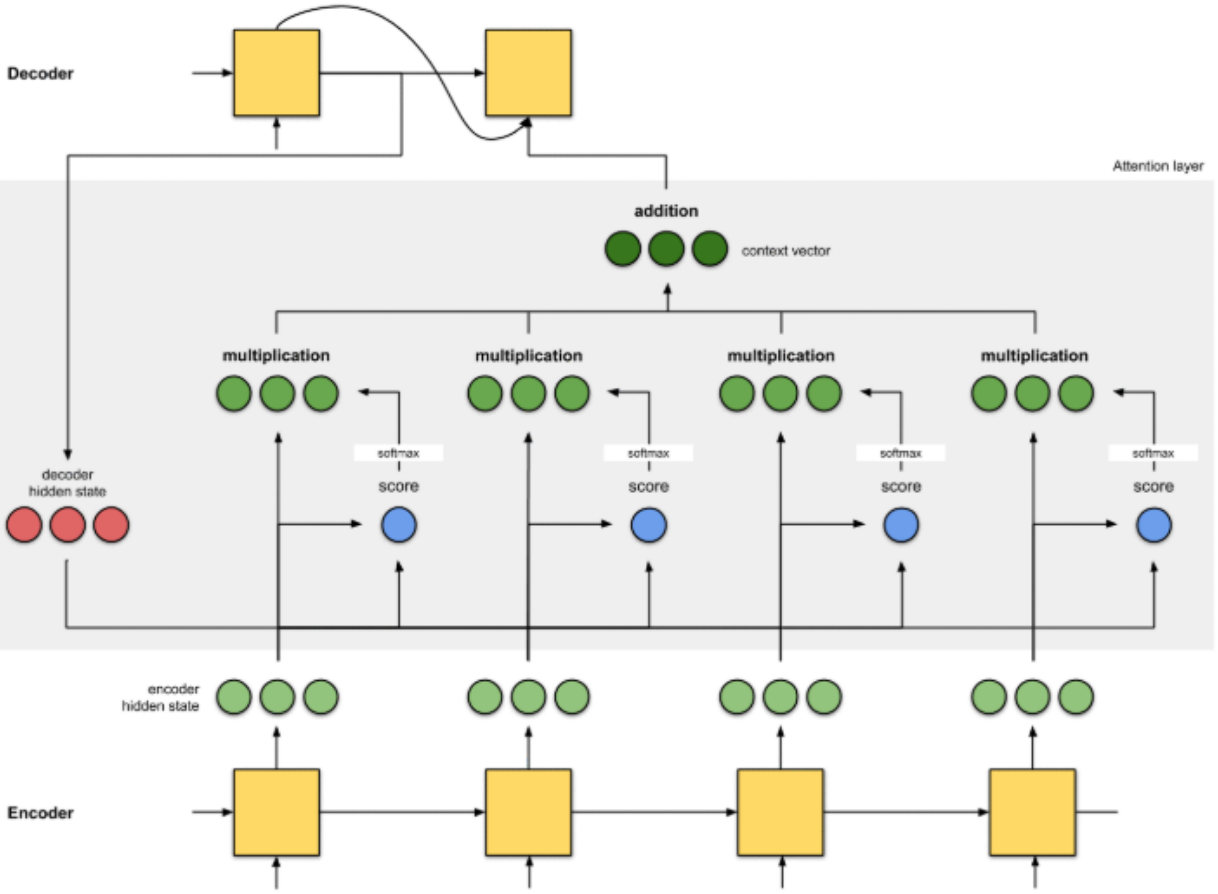
- Seq2Seq (encoder-decoder) with attention




https://machinelearningmastery.com/how-does-attention-work-in-encoder-decoder-recurrent-neural-networks/

# Transformer

- Seq2Seq (encoder-decoder) with attention




https://towardsdatascience.com/attn-illustrated-attention-5ec4ad276ee3

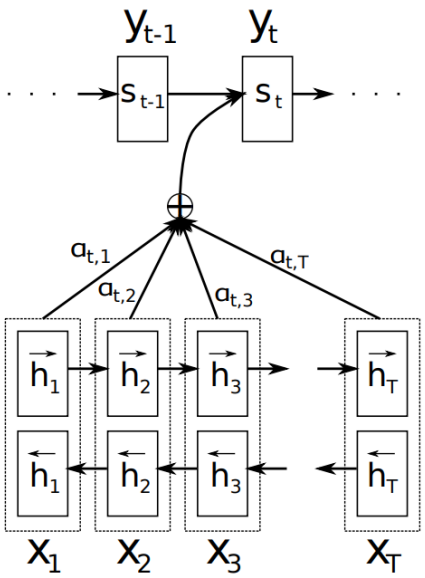
# Transformer

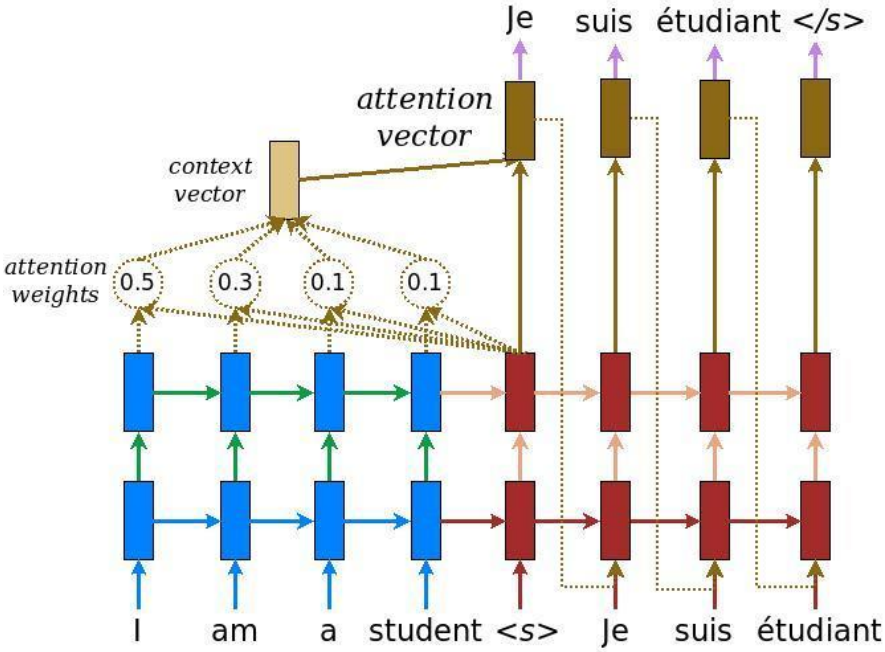
- Seq2Seq (encoder-decoder) with attention

$$\alpha_{ts} = \frac{\exp\left(\text{score}(\boldsymbol{h}_t, \bar{\boldsymbol{h}}_s)\right)}{\sum_{s'=1}^{S} \exp\left(\text{score}(\boldsymbol{h}_t, \bar{\boldsymbol{h}}_{s'})\right)} \quad \text{[Attention weights]} \quad (1)$$

$$\boldsymbol{c}_t = \sum_s \alpha_{ts} \bar{\boldsymbol{h}}_s \quad \text{[Context vector]} \quad (2)$$

$$\boldsymbol{a}_t = f(\boldsymbol{c}_t, \boldsymbol{h}_t) = \tanh(\boldsymbol{W}_{\boldsymbol{c}}[\boldsymbol{c}_t; \boldsymbol{h}_t]) \quad \text{[Attention vector]} \quad (3)$$

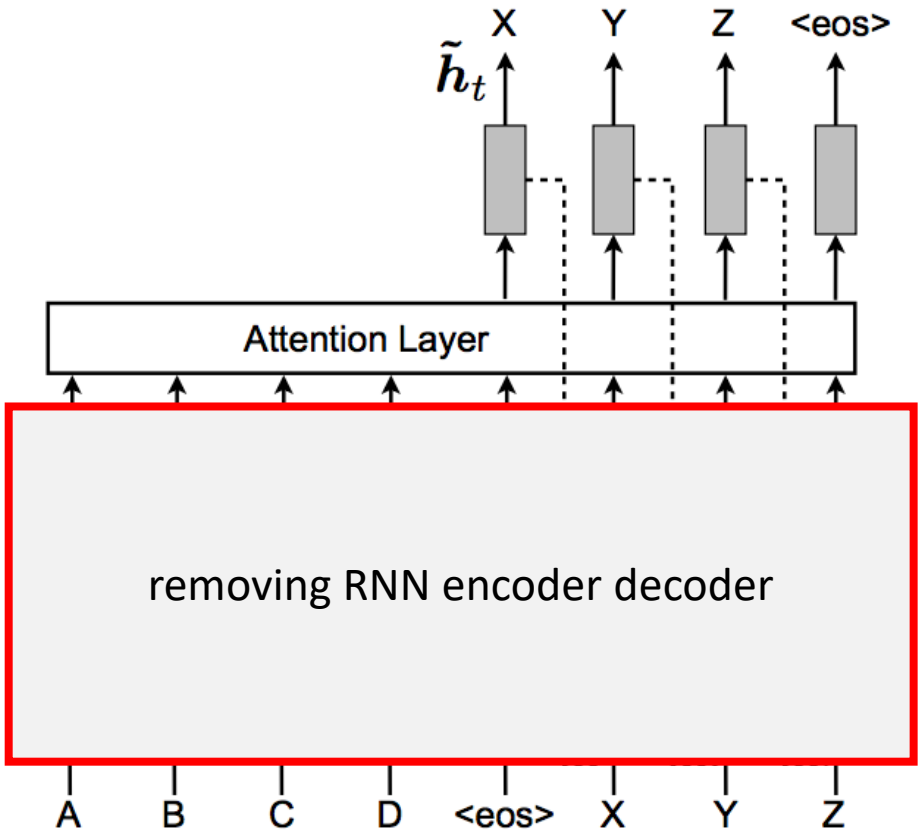



- Bahdanau, Dzmitry, Kyunghyun Cho, and Yoshua Bengio. "Neural machine translation by jointly learning to align and translate." arXiv preprint arXiv:1409.0473 (2014).
- https://www.tensorflow.org/tutorials/text/nmt_with_attention

# Transformer

- Transformer
  - Removing RNN
  - How can we obtained attention score? → self attention



# Transformer

- Transformer
  - Encoder
    - Positional embedding
    - Multi-head attention
    - Add & layer normalization
  - Decoder
    - Inputs
    - Masked multi-head attention
    - Outputs

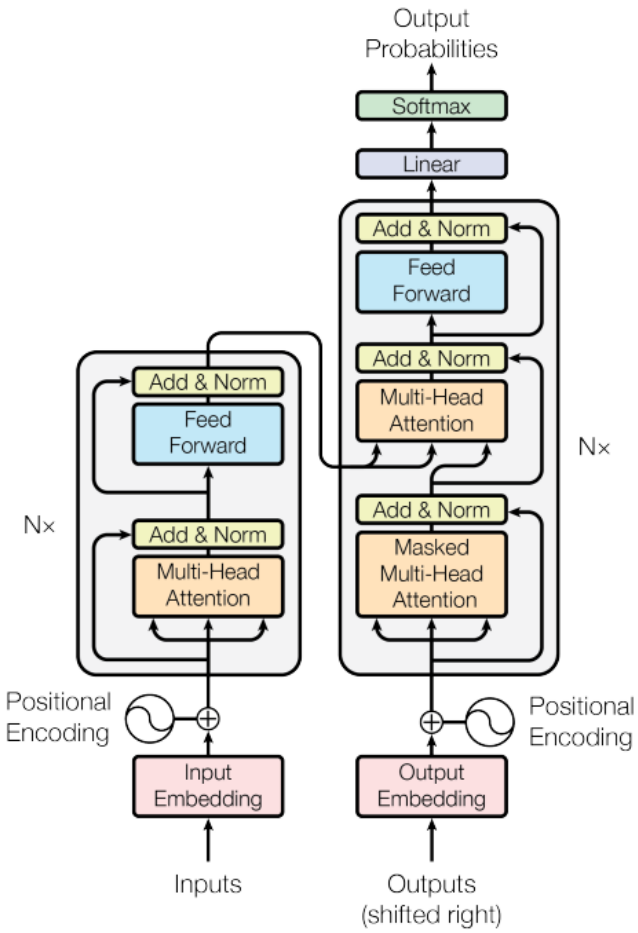


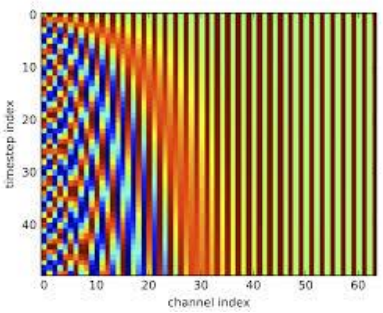

Figure 1: The Transformer - model architecture.

- Vaswani, Ashish, et al. "Attention is all you need." *Advances in neural information processing systems*. 2017.

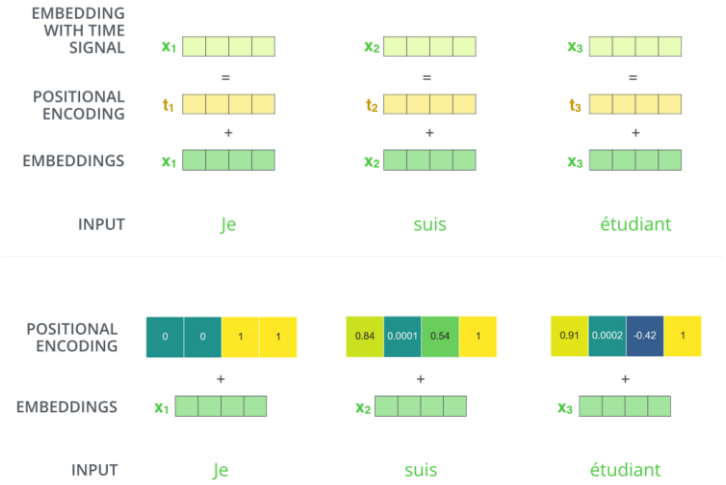
# Transformer

- Transformer
  - Encoder
    - Positional embedding

  - Actual word representations are **byte-pair encodings**
  - Also added is a positional encoding so same words **at different locations have different overall representations**




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



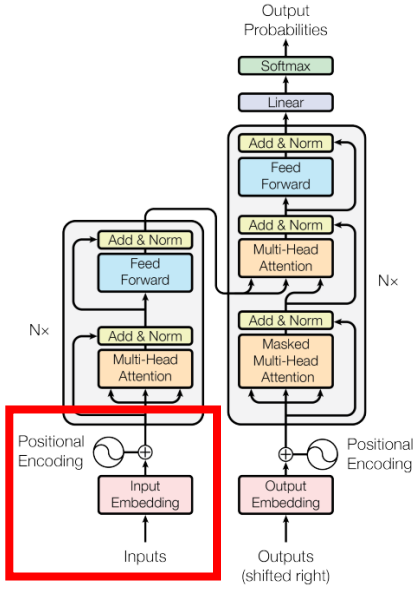
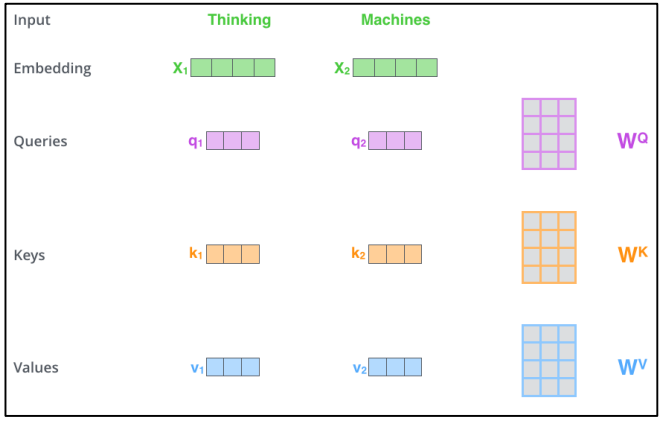



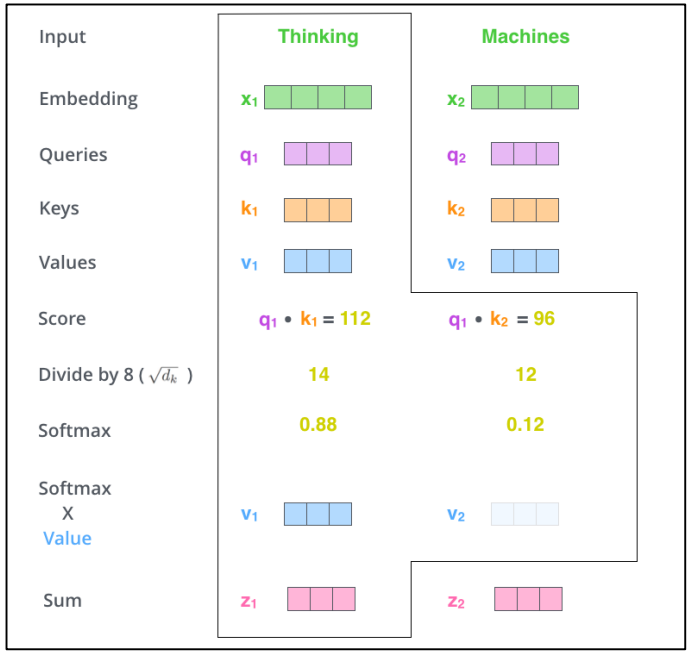

Figure 1: The Transformer - model architecture.

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

# Transformer

- Transformer
  - Encoder
    - Multi-head attention
  - Self-attention
    - Input: embeddings
    - Weights: Wq, Wk, Wv
    - Queries, keys values



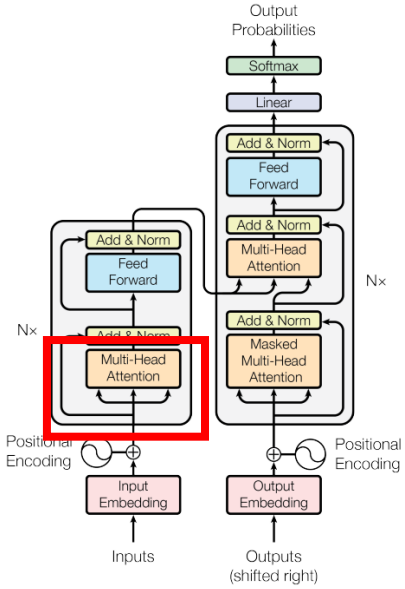
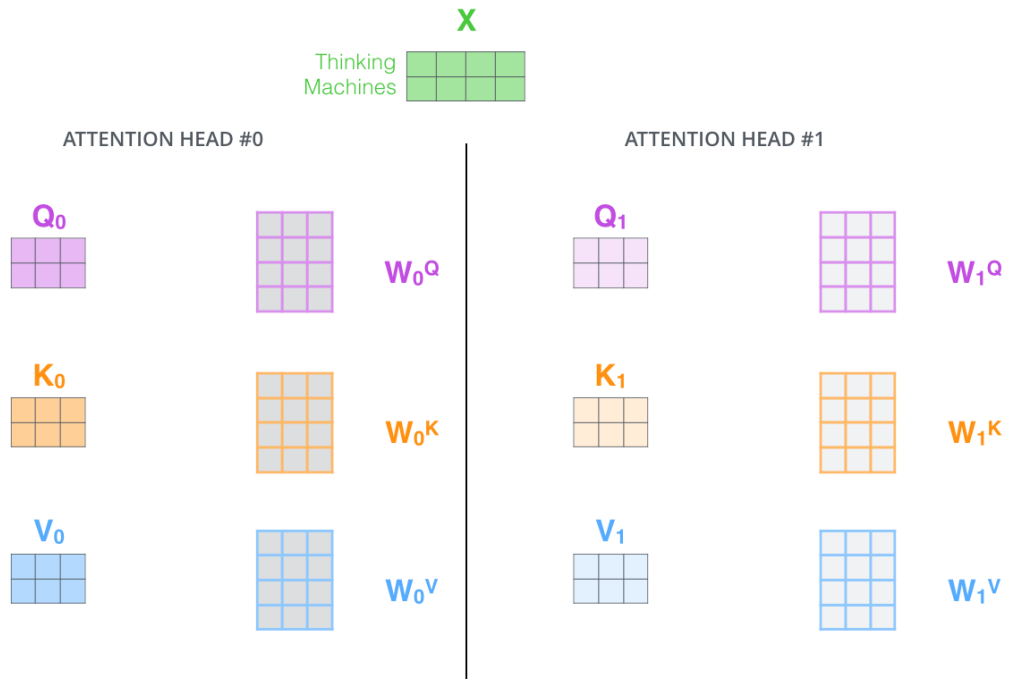




Figure 1: The Transformer - model architecture.

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

# Transformer

- Transformer
  - Encoder
    - Multi-head attention
  - Multi-head



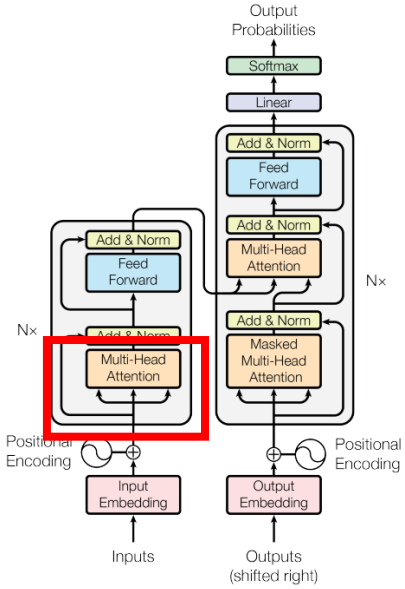
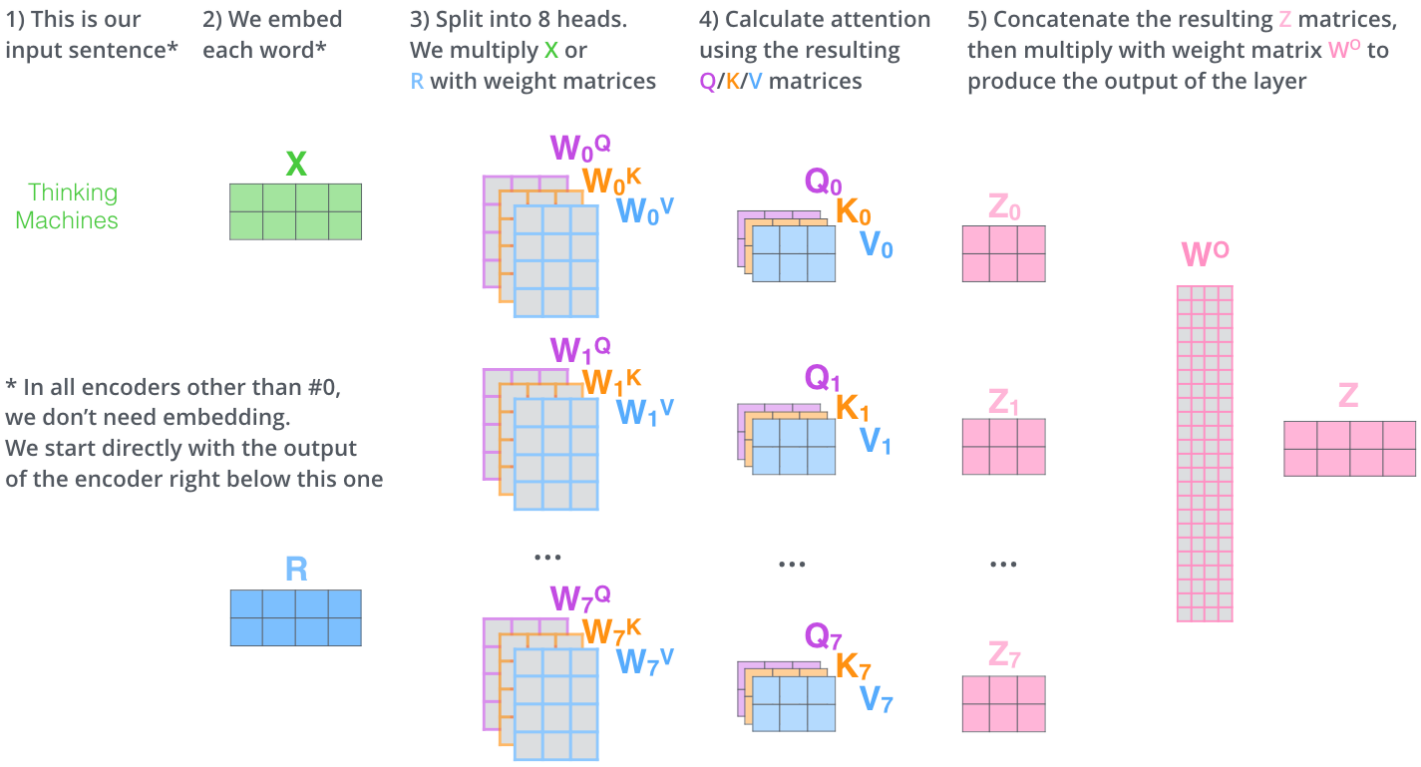




Figure 1: The Transformer - model architecture.

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

# Transformer

- Transformer
  - Encoder
    - Multi-head attention



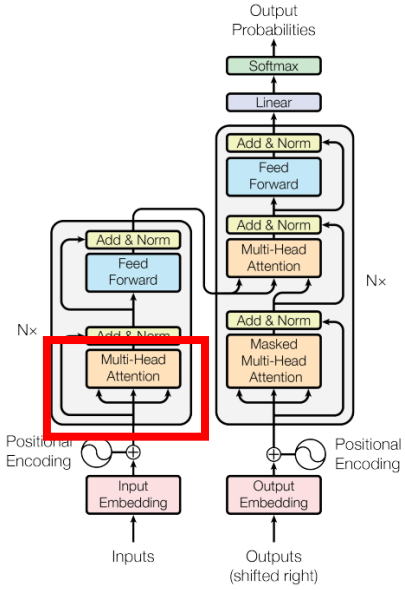
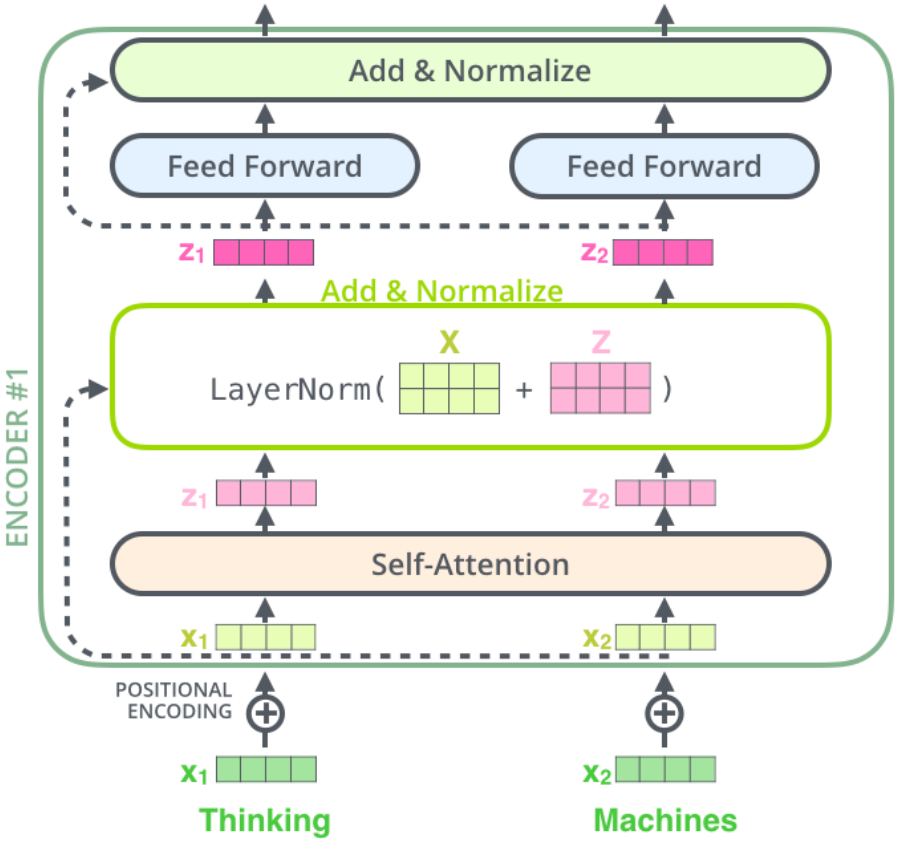




Figure 1: The Transformer - model architecture.

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

# Transformer

- Transformer
  - Encoder
    - Add & layer normalization



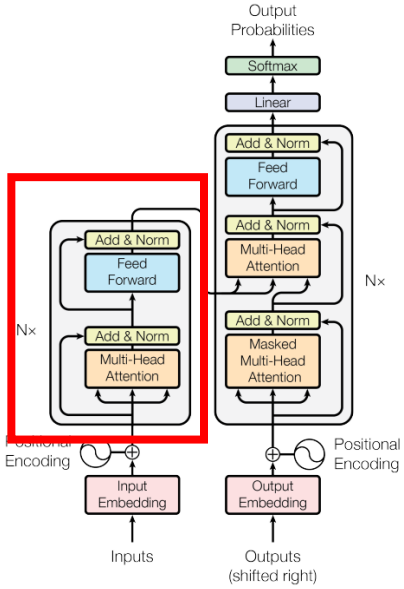
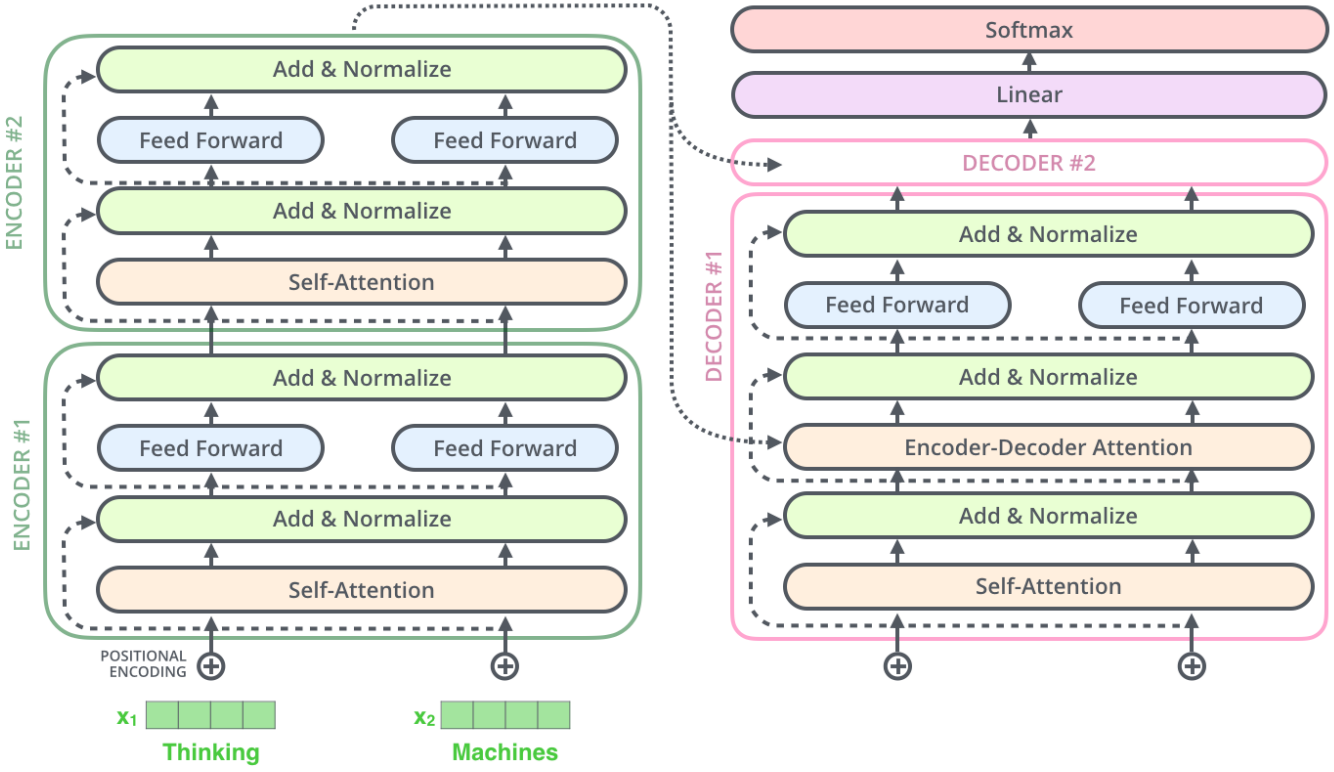




Figure 1: The Transformer - model architecture.

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

# Transformer

- Transformer
  - Encoder
    - Add & layer normalization



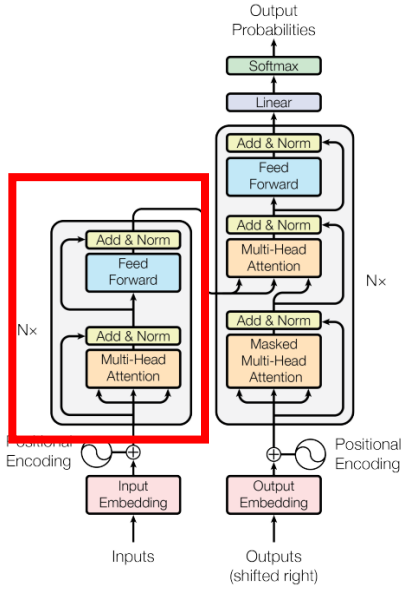
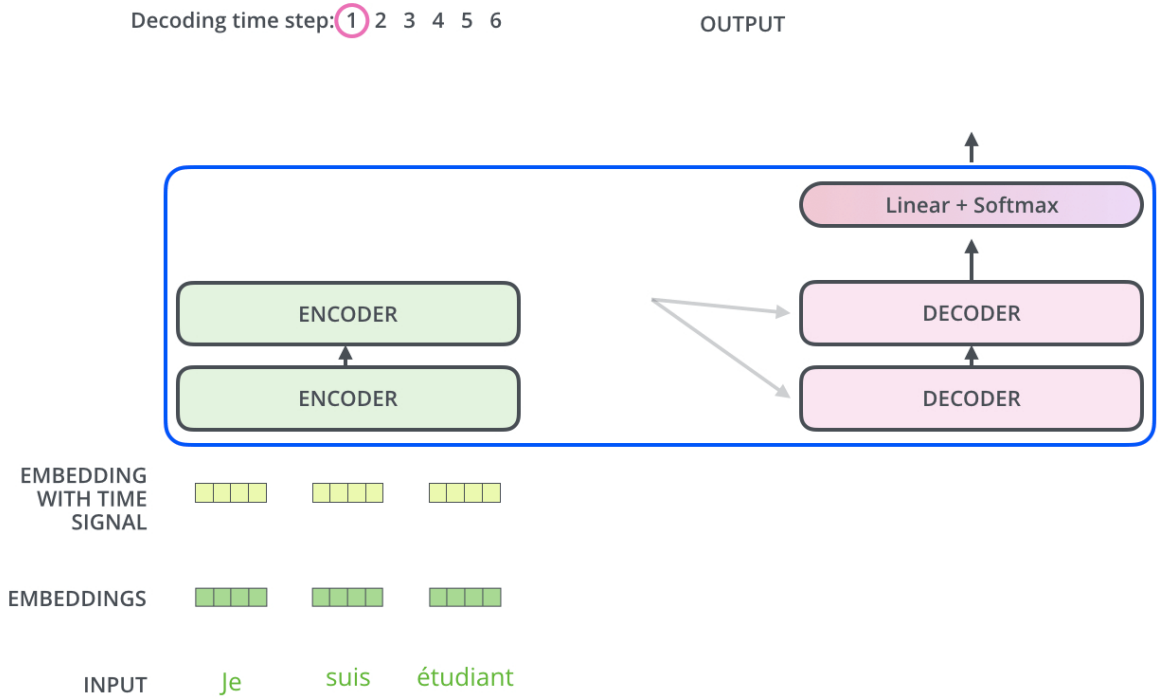




Figure 1: The Transformer - model architecture.

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

# Transformer

- Transformer
  - Decoder
    - Inputs : Key(enc), Value(enc), Word embeddings (dec)



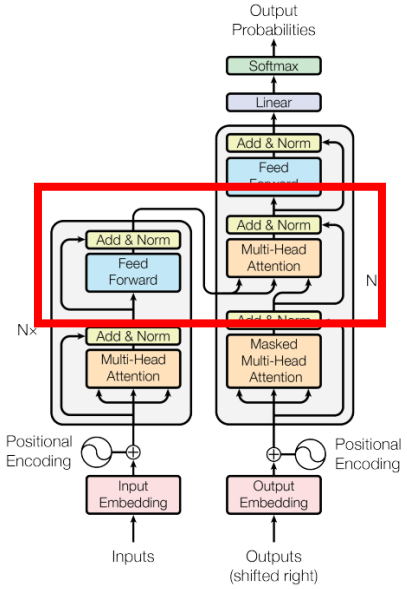
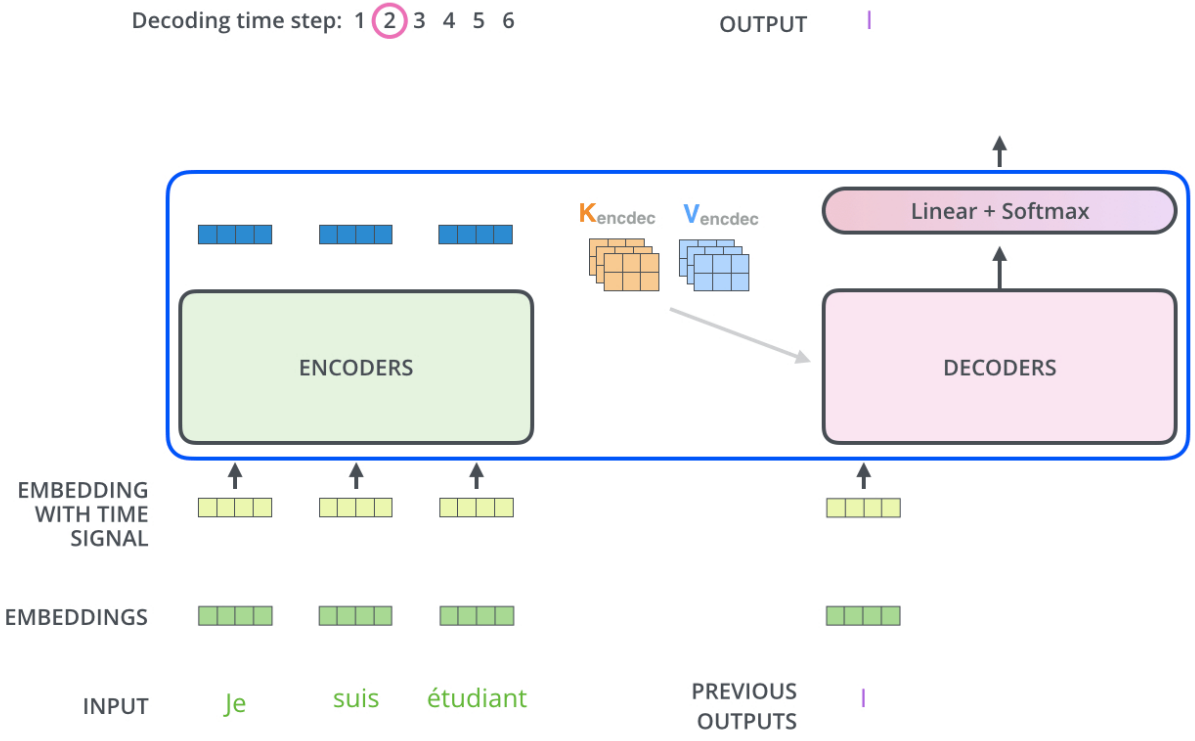




Figure 1: The Transformer - model architecture.

# Transformer

- Transformer
  - Decoder
    - Inputs



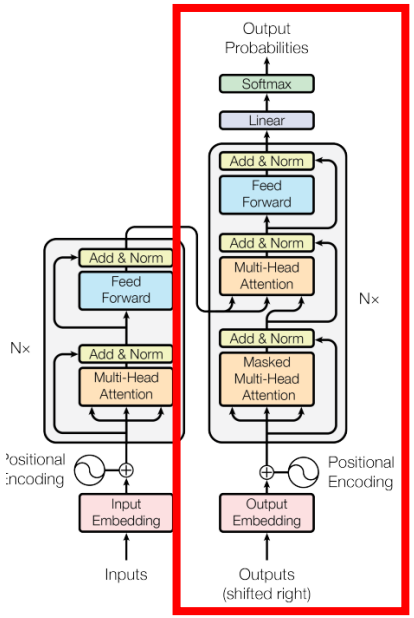




Figure 1: The Transformer - model architecture.

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

# Transformer

- Transformer
  - Encoder
    - Positional embedding
    - Multi-head attention
    - Add & layer normalization
  - Decoder
    - Masked multi-head attention
    - Multi-head attention

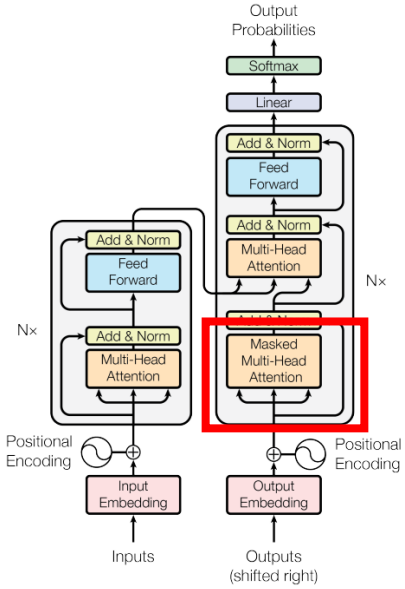
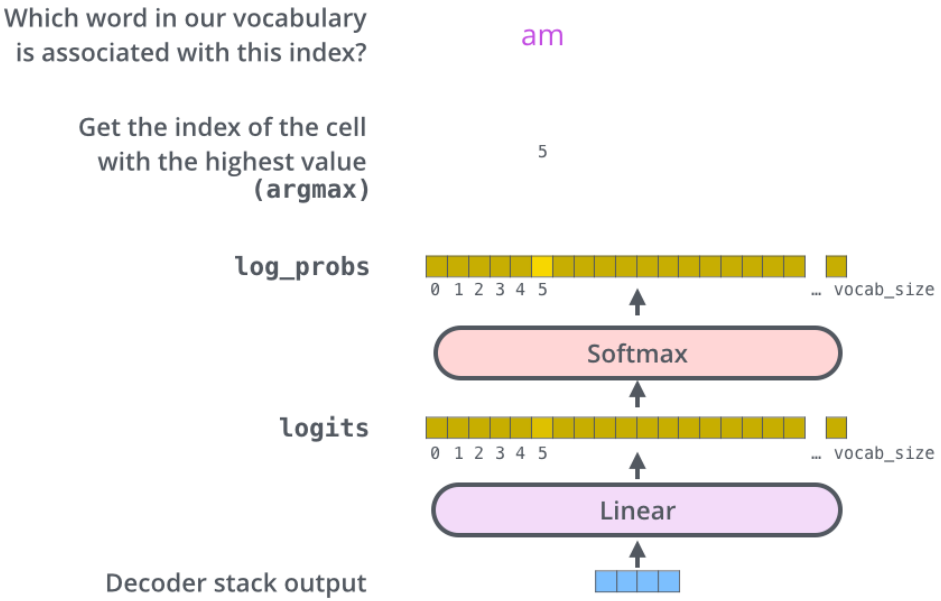



Figure 1: The Transformer - model architecture.

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

# Transformer

- Transformer
  - Decoder
    - Outputs
      - Cross-entropy, KL Divergence



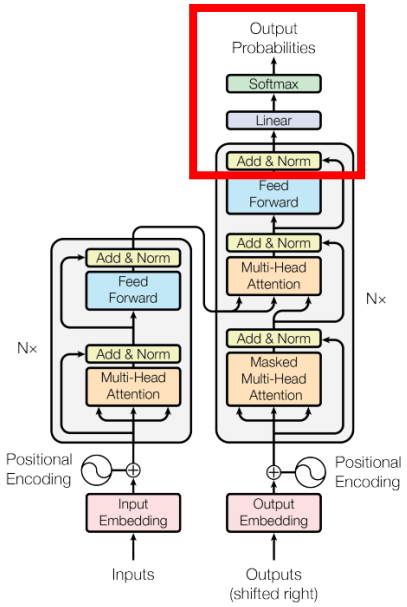




Figure 1: The Transformer - model architecture.

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

# Transformer

- Experiments

Table 2: The Transformer achieves better BLEU scores than previous state-of-the-art models on the English-to-German and English-to-French newstest2014 tests at a fraction of the training cost.

| Model | BLEU | | Training Cost (FLOPs) | |
|---|---|---|---|---|
| | EN-DE | EN-FR | EN-DE | EN-FR |
| ByteNet [18] | 23.75 | | | |
| Deep-Att + PosUnk [39] | | 39.2 | | $1.0 \cdot 10^{20}$ |
| GNMT + RL [38] | 24.6 | 39.92 | $2.3 \cdot 10^{19}$ | $1.4 \cdot 10^{20}$ |
| ConvS2S [9] | 25.16 | 40.46 | $9.6 \cdot 10^{18}$ | $1.5 \cdot 10^{20}$ |
| MoE [32] | 26.03 | 40.56 | $2.0 \cdot 10^{19}$ | $1.2 \cdot 10^{20}$ |
| Deep-Att + PosUnk Ensemble [39] | | 40.4 | | $8.0 \cdot 10^{20}$ |
| GNMT + RL Ensemble [38] | 26.30 | 41.16 | $1.8 \cdot 10^{20}$ | $1.1 \cdot 10^{21}$ |
| ConvS2S Ensemble [9] | 26.36 | **41.29** | $7.7 \cdot 10^{19}$ | $1.2 \cdot 10^{21}$ |
| Transformer (base model) | 27.3 | 38.1 | **$3.3 \cdot 10^{18}$** | |
| Transformer (big) | **28.4** | **41.8** | $2.3 \cdot 10^{19}$ | |

- Base model for other language models
  - BERT, GPT …


- Vaswani, Ashish, et al. "Attention is all you need." *Advances in neural information processing systems*. 2017.

# Reference


- Bengio, Yoshua, et al. "A neural probabilistic language model." *Journal of machine learning research* 3.Feb (2003): 1137-1155.
- Mikolov, Tomas, et al. "Distributed representations of words and phrases and their compositionality." Advances in neural information processing systems. 2013.
- https://wikidocs.net/22660
- https://dengliangshi.github.io/2017/01/01/neural-network-language-models.html
- Peters, Matthew E., et al. "Deep contextualized word representations." arXiv preprint arXiv:1802.05365 (2018).
- https://www.topbots.com/generalized-language-models-cove-elmo/
- https://medium.com/@plusepsilon/the-bidirectional-language-model-1f3961d1fb27
- https://wikidocs.net/33930
- https://towardsdatascience.com/seq2seq-model-in-tensorflow-ec0c557e560f
- Cho, Kyunghyun, et al. "On the properties of neural machine translation: Encoder-decoder approaches." arXiv preprint arXiv:1409.1259 (2014).
- https://machinelearningmastery.com/how-does-attention-work-in-encoder-decoder-recurrent-neural-networks/
- https://towardsdatascience.com/attn-illustrated-attention-5ec4ad276ee3
- Bahdanau, Dzmitry, Kyunghyun Cho, and Yoshua Bengio. "Neural machine translation by jointly learning to align and translate." arXiv preprint arXiv:1409.0473 (2014).
- https://www.tensorflow.org/tutorials/text/nmt_with_attention
- Vaswani, Ashish, et al. "Attention is all you need." Advances in neural information processing systems. 2017.
- http://jalammar.github.io/illustrated-transformer/

# Q&A
# Thank you!