Self Attention & BERT using TensorFlow 2.0

Souradip Chakraborty - Data Scientist @Walmart Labs
Indian Statistical Institute

TensorFlow User Group KOLKATA

Google Developer Expert

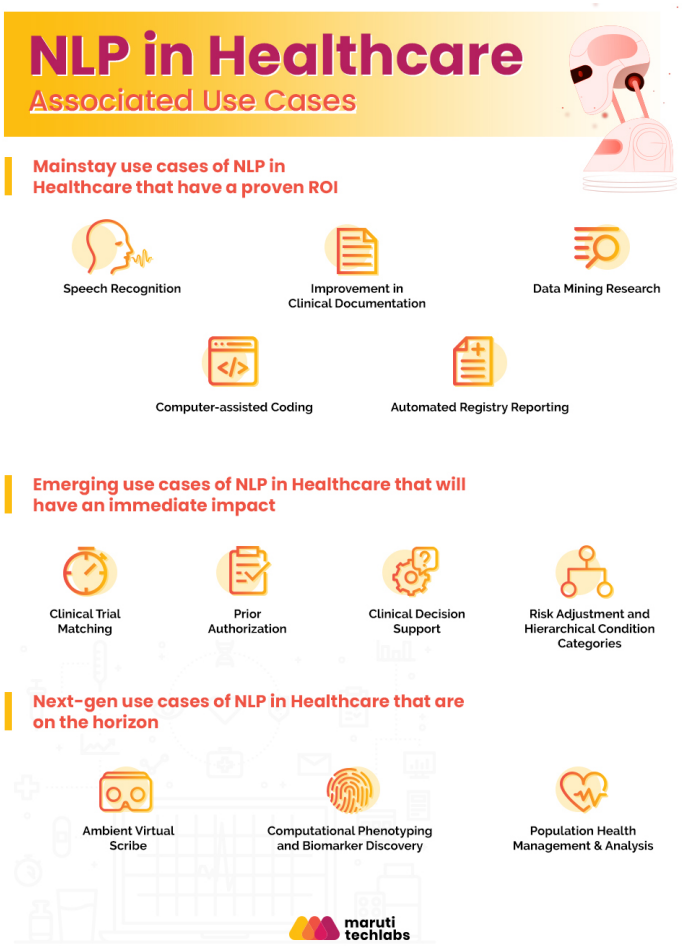
TensorFlow 2.0

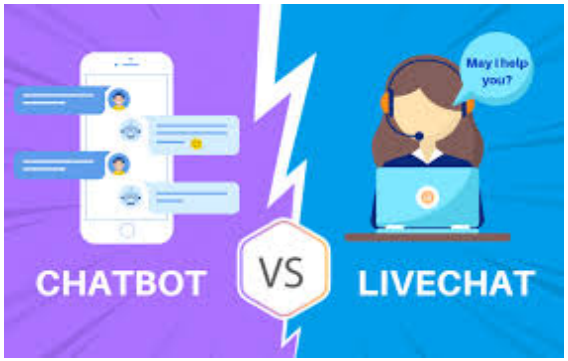
Session Agenda

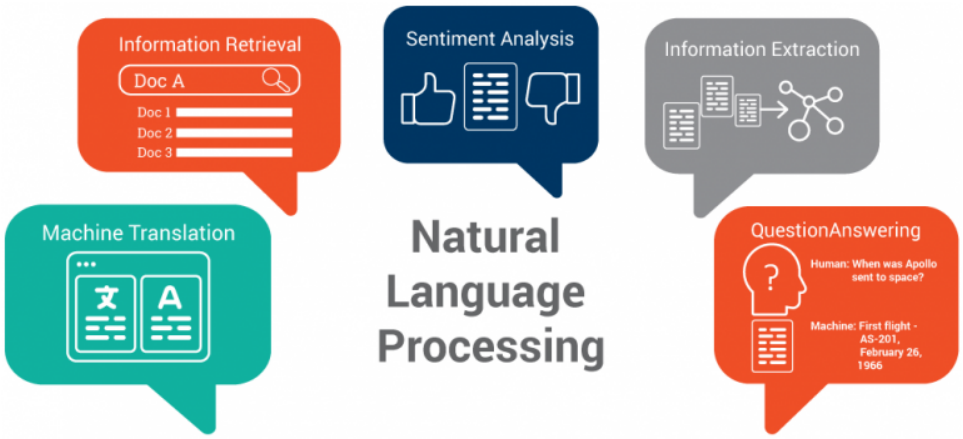
TensorFlow User Group KOLKATA

- Intro to Natural Language Processing and Application Domains
- History & Evolution of NLP Embeddings
- Sequence Learning Tasks and Recurrent Neural Network(RNNs & LSTMs)
- Introduction to Attention Mechanism – Attention is all you need !!
- Self Attention (Single & Multiheaded) & Transformer Architecture
- State of the Art Language Model for NLP - BERT
- Code walk-through in TensorFlow 2.0

Natural Language Processing and Application Domains







History & Evolution of NLP Embeddings – Bag of Words approach

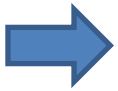
1. **Term Frequency Vectorizer** - It only takes into consideration the frequency of words in a document. A document's important words are words that occur a lot of times in said document.

Suppose our set of documents consists of three sentences:

1. I love dogs.

2. I hate dogs and knitting.

3. Knitting is my hobby and my passion.



	I	love	dogs	hate	and	knitting	is	my	hobby	passion
Doc 1	1	1	1							
Doc 2	1		1	1	1	1				
Doc 3					1	1	1	2	1	1

History & Evolution of NLP Embeddings – Bag of Words approach

2. **Term Frequency Inverse Document Vectorizer** - It takes into consideration the frequency of words in a document as well as the occurrence of that word in other documents. A document's important words are words that occur a lot of times in said document and must be absent in the other documents .

$$w_{x,y} = tf_{x,y} \times \log\left(\frac{N}{df_x}\right)$$

TF-IDF

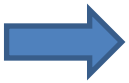
Term x within document y

$tf_{x,y}$ = frequency of x in y
df_x = number of documents containing x
N = total number of documents

1. I love dogs.

2. I hate dogs and knitting.

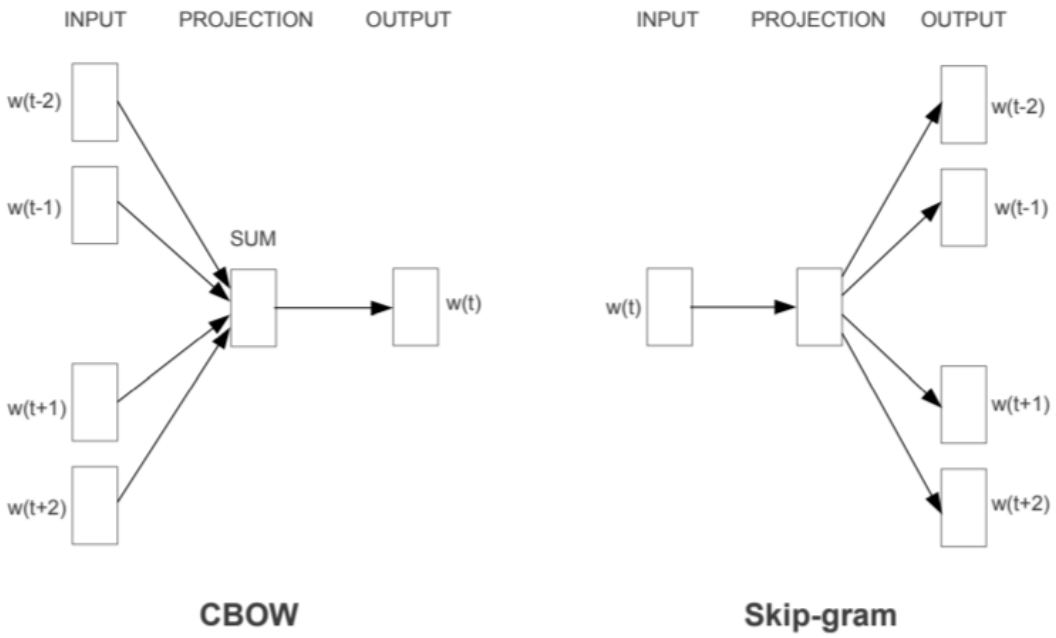
3. Knitting is my hobby and my passion.



	I	love	dogs	hate	and	knitting	is	my	hobby	passion
Doc 1	0.18	**0.48**	0.18							
Doc 2	0.18		0.18	**0.48**	0.18	0.18				
Doc 3					0.18	0.18	**0.48**	**0.95**	**0.48**	**0.48**

History & Evolution of NLP Embeddings – Co-occurrence approach

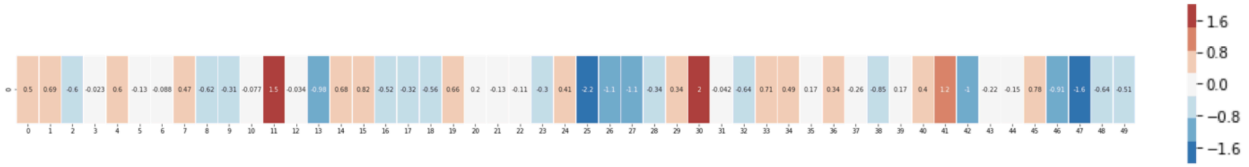
Continuous Bag of Words Model (CBOW) and Skip-gram



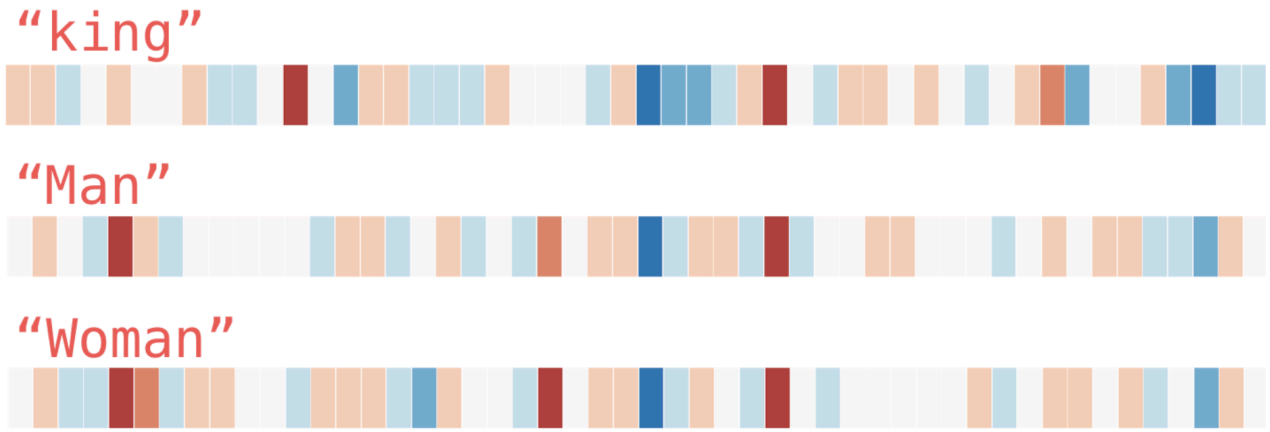
In the CBOW model, the distributed representations of context (or surrounding words) are combined to predict the word in the middle. While in the Skip-gram model, the distributed representation of the input word is used to predict the context.

Glove Embeddings (Trained on Wikipedia) – Visual Explanation

Let's color code the cells based on their values (red if they're close to 2, white if they're close to 0, blue if they're close to -2):



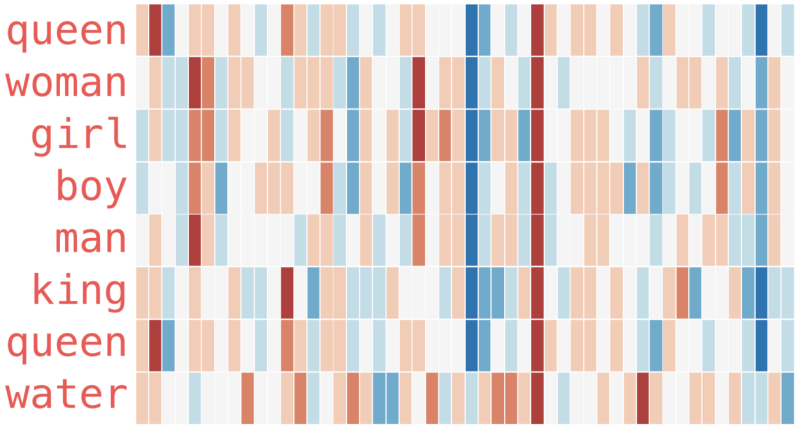
We'll proceed by ignoring the numbers and only looking at the colors to indicate the values of the cells. Let's now contrast "King" against other words:

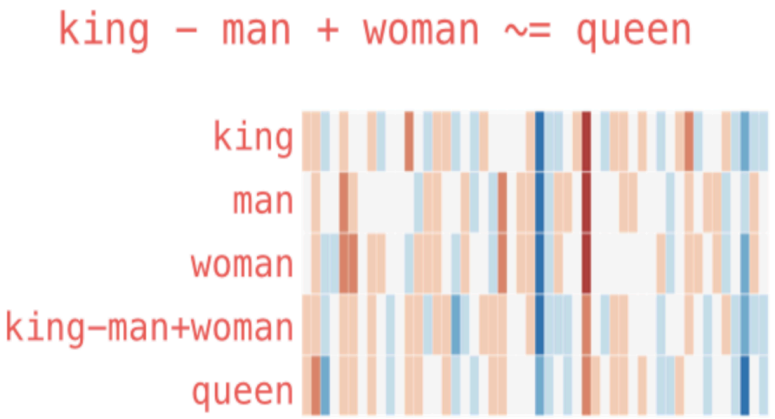


Ref : http://jalammar.github.io/illustrated-word2vec/

History & Evolution of NLP Embeddings – Word Embeddings

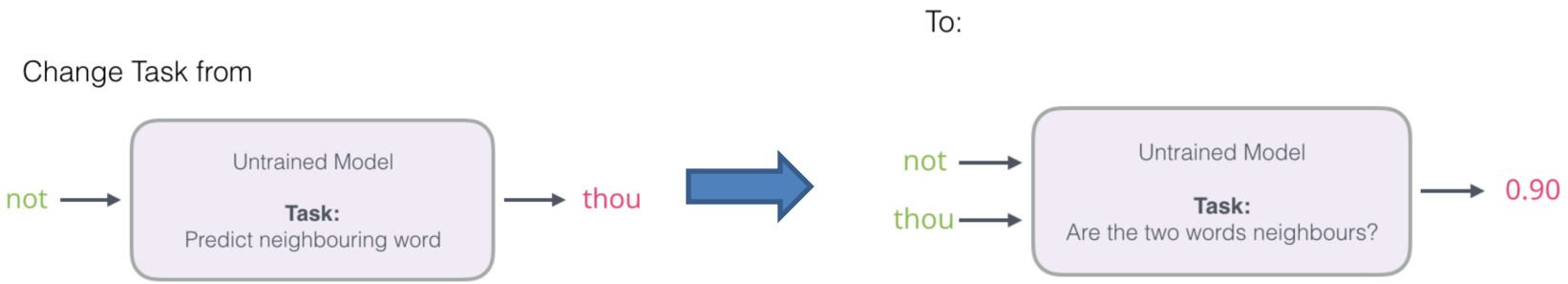
Glove Embeddings (Trained on Wikipedia) – Visual Explanation





Ref : http://jalammar.github.io/illustrated-word2vec/

History & Evolution of NLP Embeddings – Negative Sampling



input word	target word
not	thou
not	shalt
not	make
not	a
make	shalt
make	not
make	a
make	machine

input word	output word	target
not	thou	1
not	shalt	1
not	make	1
not	a	1
make	shalt	1
make	not	1
make	a	1
make	machine	1

Ref : http://jalammar.github.io/illustrated-word2vec/

History & Evolution of NLP Embeddings – Negative Sampling

Problem : One major drawback of the approach is if we don't have negative samples, model will generate useless embeddings as it is a trivial problem of single class.

Negative Sample Mining : To solve the problem, we introduce *negative samples* to our dataset – samples of words that are not neighbours, which will be labelled as 0.

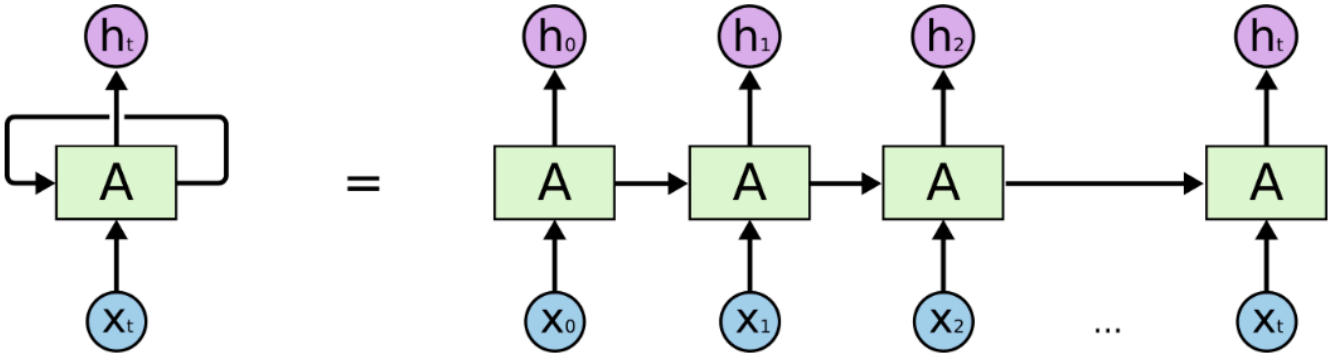
Ref : http://jalammar.github.io/illustrated-word2vec/

Sequence Learning Tasks and Recurrent Neural Network(RNNs & LSTMs)

Recurrent Neural Network : The inherent sequential nature of the Natural language is being captured by RNNs.

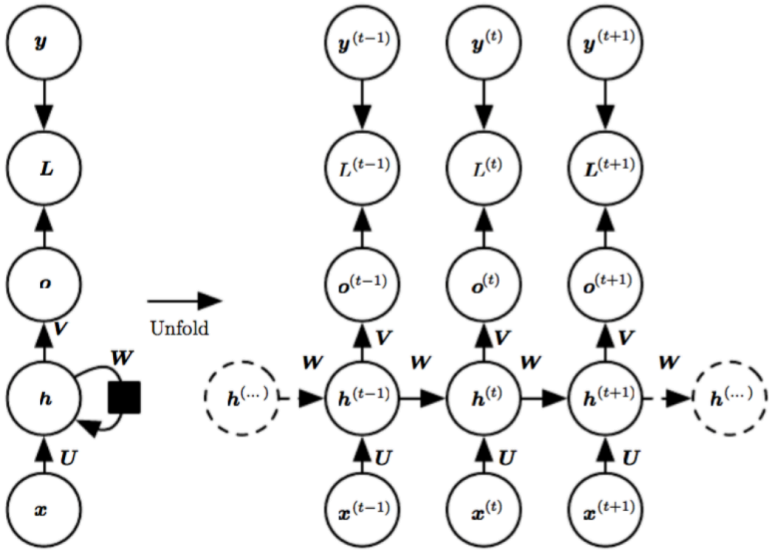
Humans don't start their thinking from scratch every second. As you read this essay, you understand each word based on your understanding of previous words. You don't throw everything away and start thinking from scratch again. Your thoughts have persistence.

Traditional neural networks can't do this, and they don't have the power to use the sequence/dependencies of previous state



Ref : https://colah.github.io/posts/2015-08-Understanding-LSTMs/

Sequence Learning Tasks and Recurrent Neural Network(RNNs & LSTMs)



$$
\begin{aligned}
\boldsymbol{a}^{(t)} &= \boldsymbol{b} + \boldsymbol{W}\boldsymbol{h}^{(t-1)} + \boldsymbol{U}\boldsymbol{x}^{(t)} \\
\boldsymbol{h}^{(t)} &= \tanh(\boldsymbol{a}^{(t)}) \\
\boldsymbol{o}^{(t)} &= \boldsymbol{c} + \boldsymbol{V}\boldsymbol{h}^{(t)} \\
\hat{\boldsymbol{y}}^{(t)} &= \operatorname{softmax}(\boldsymbol{o}^{(t)})
\end{aligned}
$$

Loss computation : $L(\{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(\tau)}\}, \{\boldsymbol{y}^{(1)}, \ldots, \boldsymbol{y}^{(\tau)}\}) = \sum_t L^{(t)} = \sum_t -\log \hat{y}^{(t)}_{y^{(t)}}.$

Introduction to Attention Mechanism – Attention is all you need !!

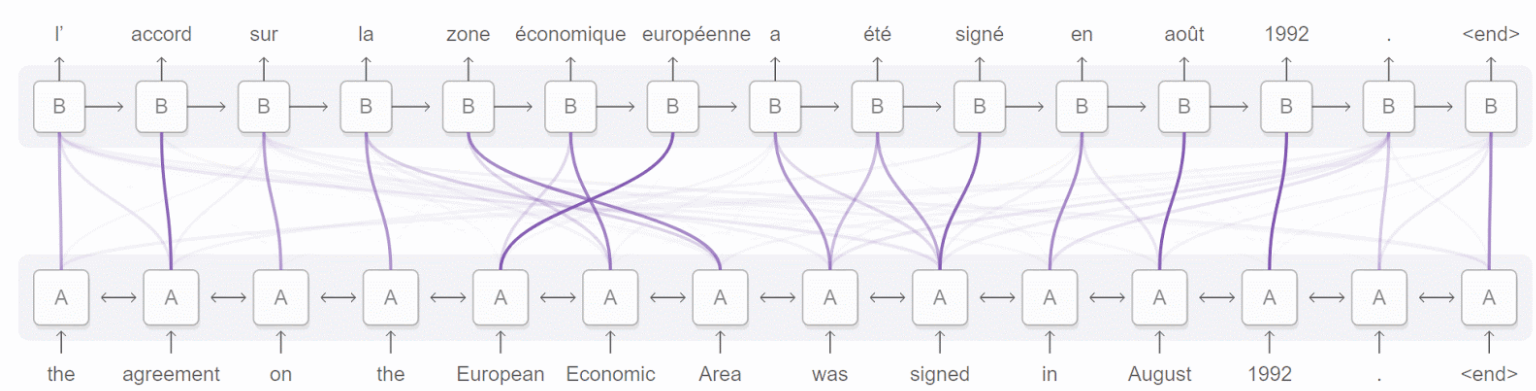
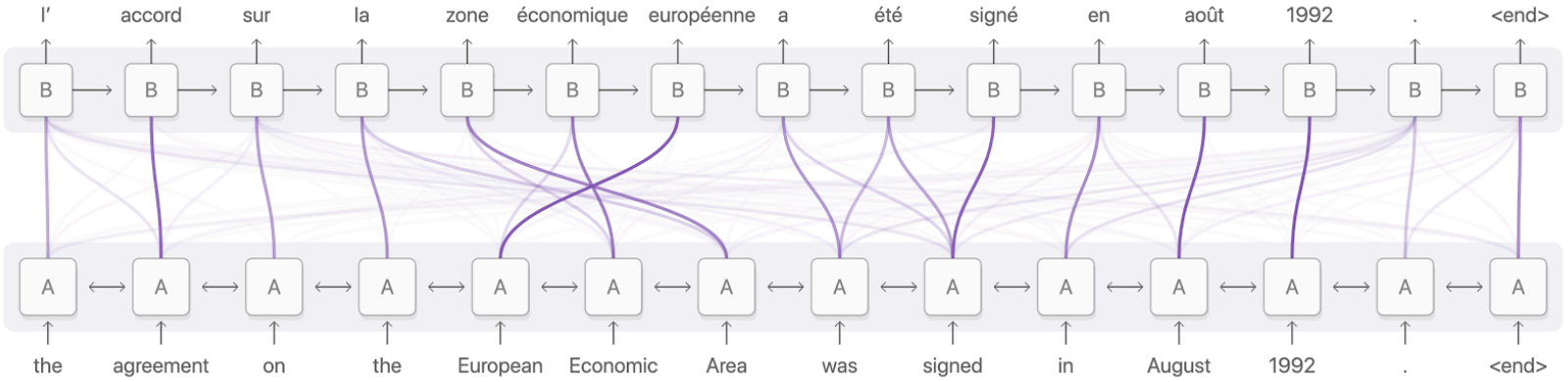
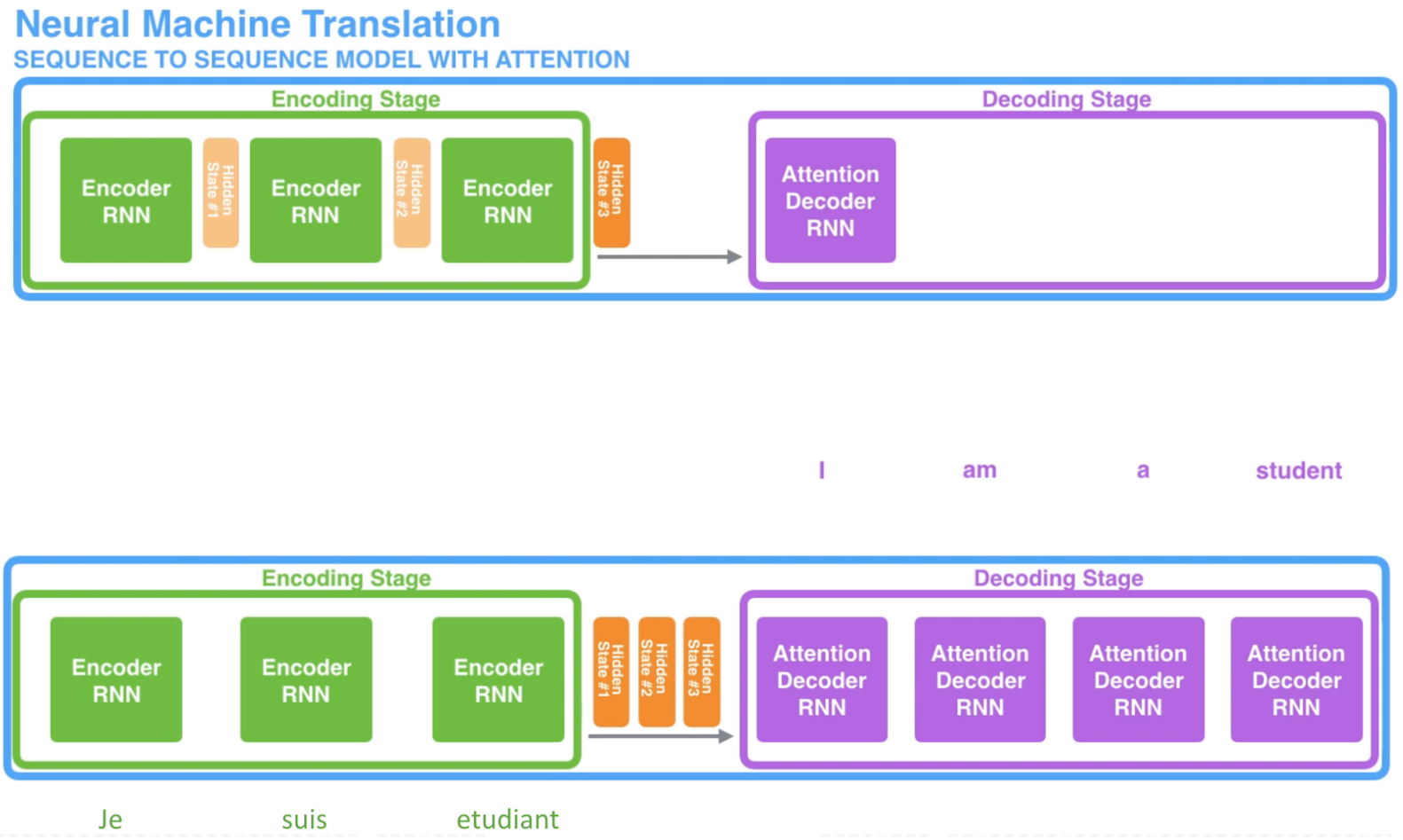


Diagram derived from Fig. 3 of Bahdanau, *et al.* 2014



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

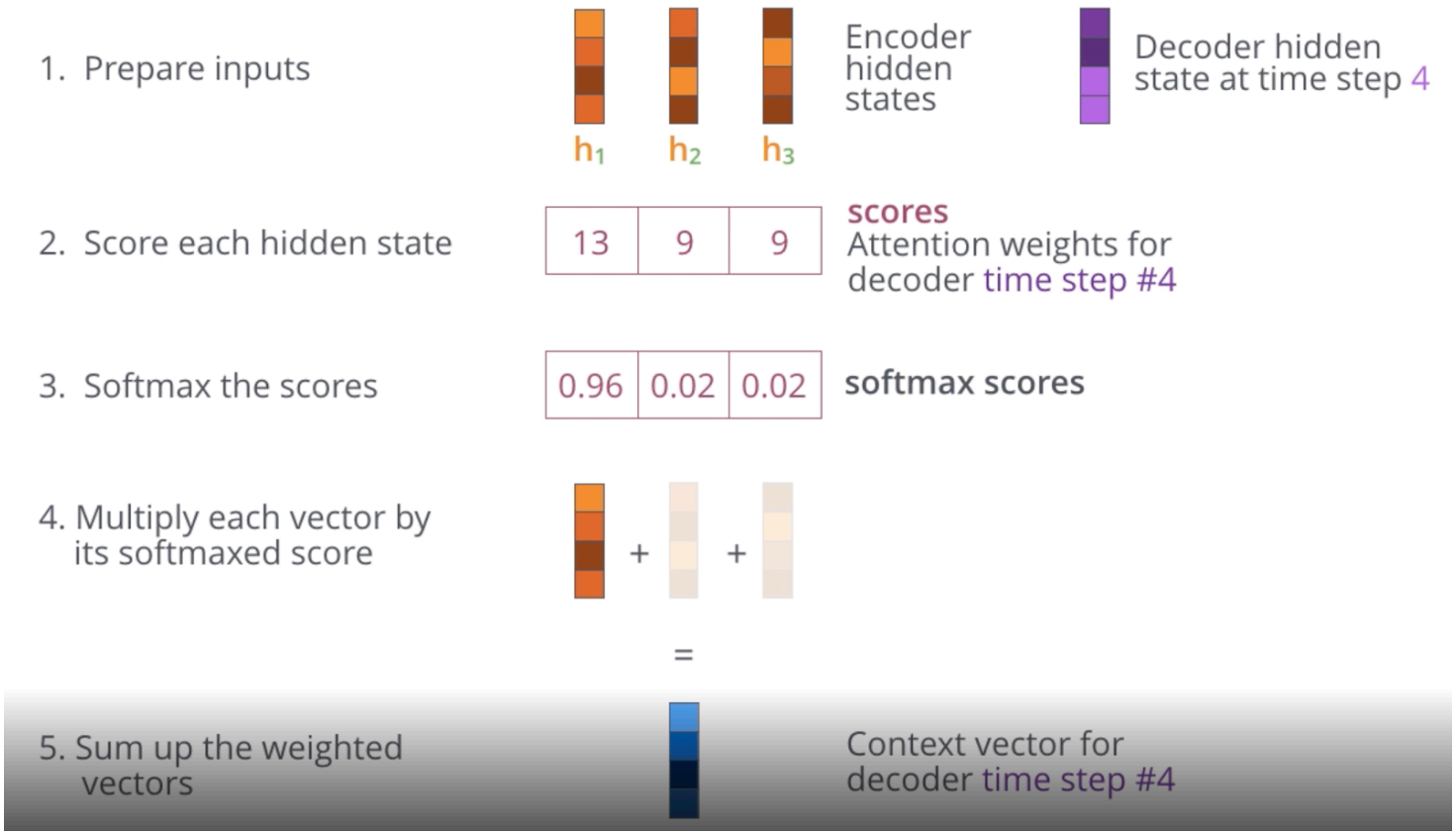
Sequence to Sequence Model with Attention



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

Introduction to Attention Mechanism – Obtaining Context Vector using Attention

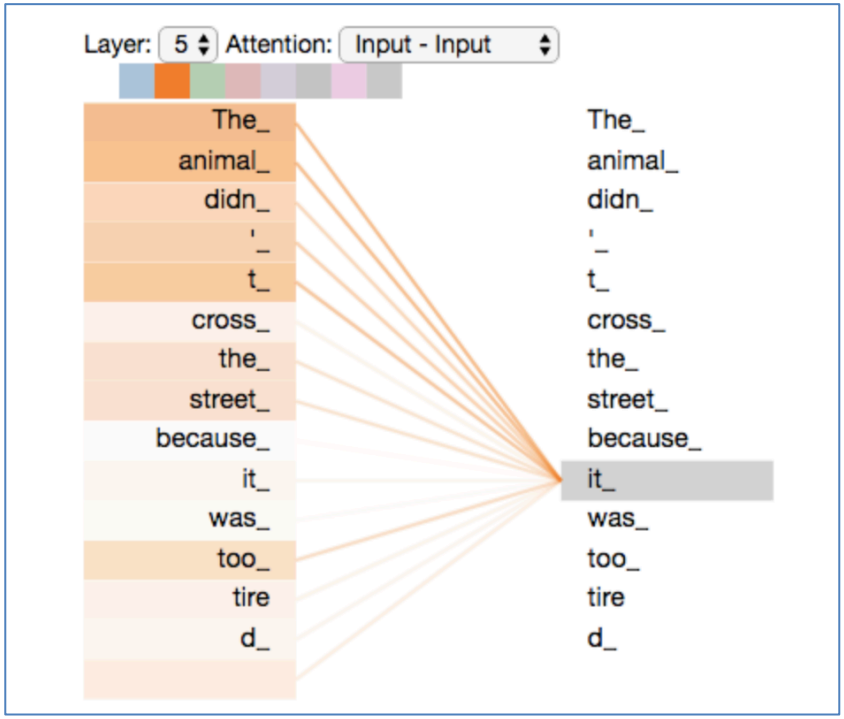
Global Attention



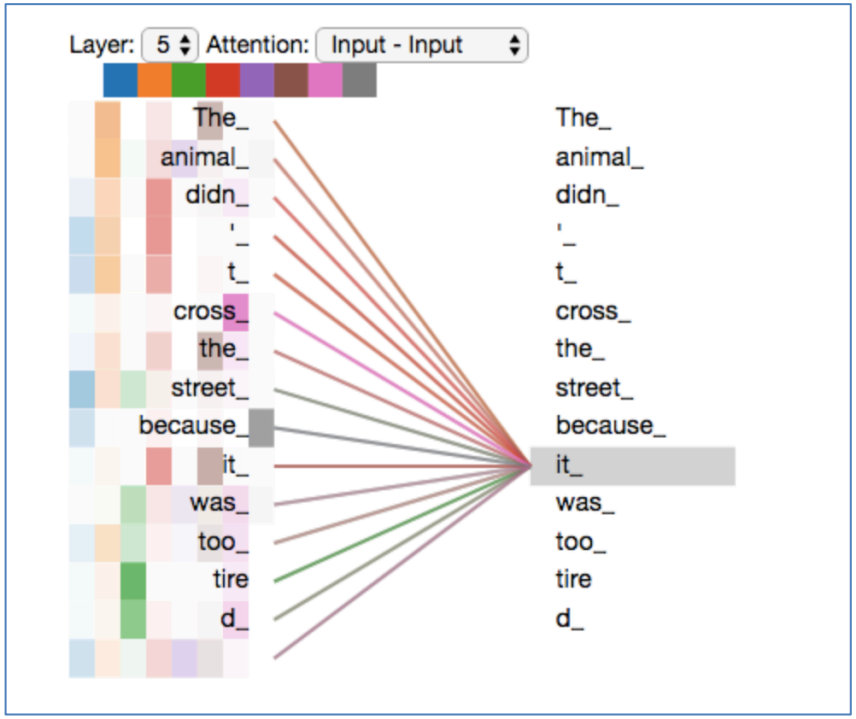
Ref : http://jalammar.github.io/illustrated-transformer/

Self Attention & Transformers

Single Head Self Attention Block

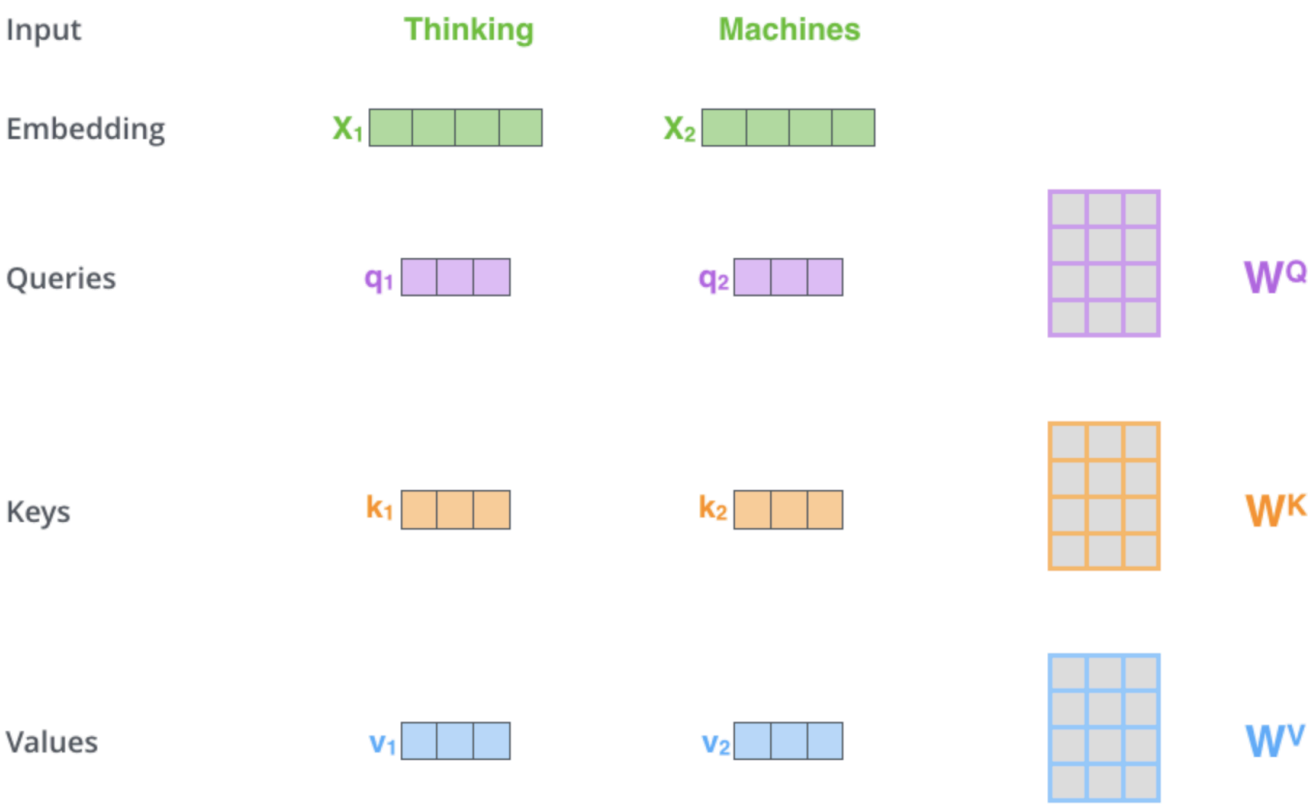


Multi Head Self Attention Block



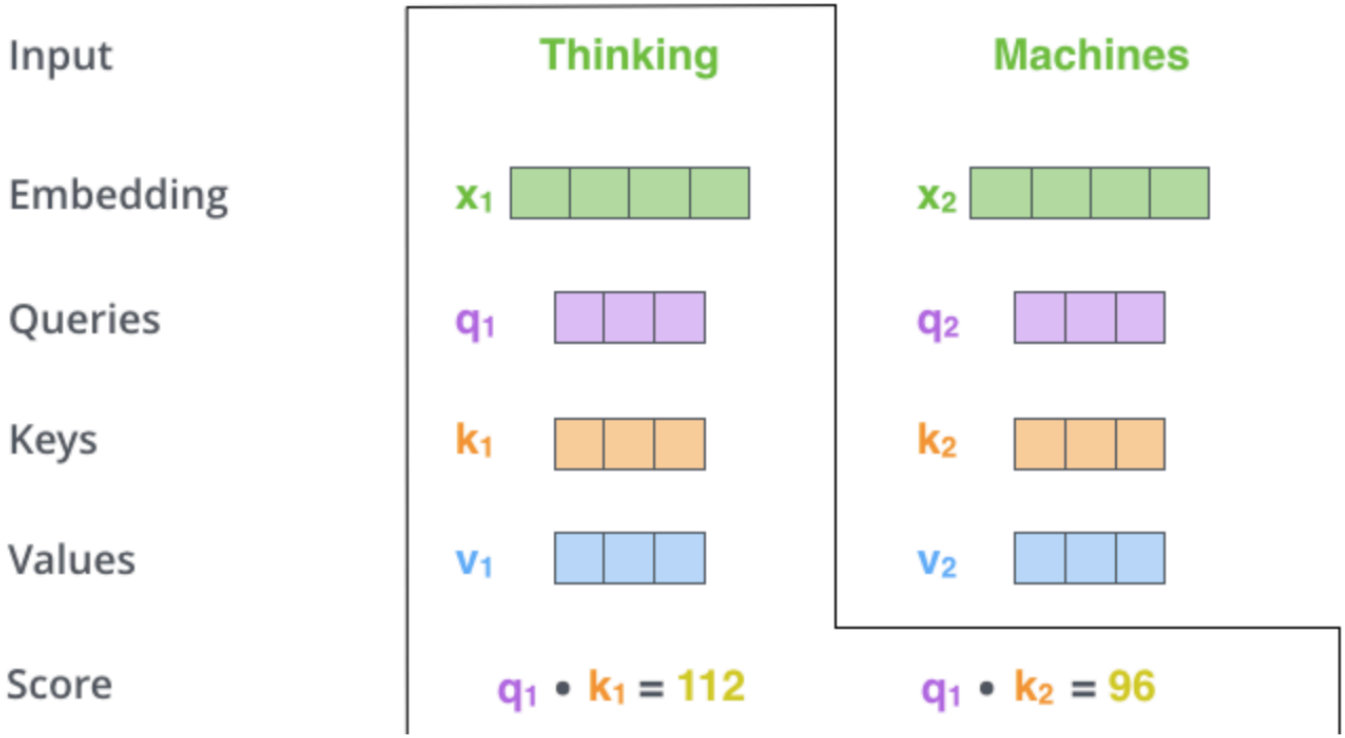
Ref : http://jalammar.github.io/illustrated-transformer/

Self Attention & Transformers - Steps



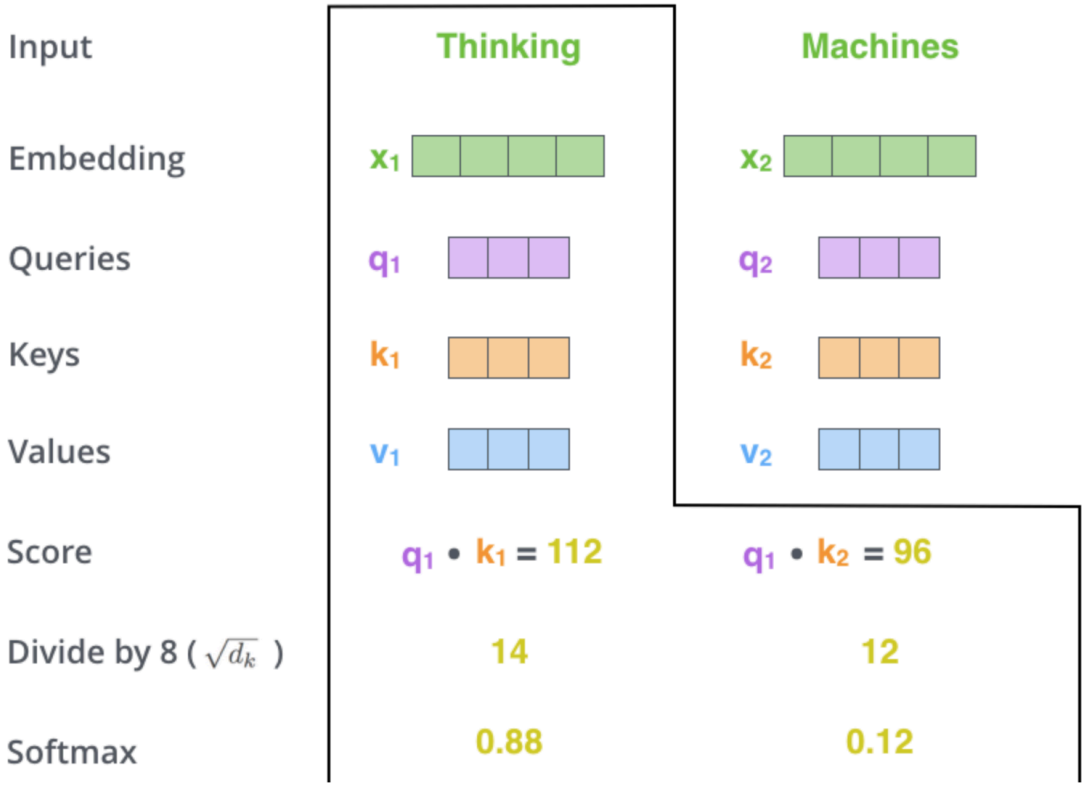
Ref : http://jalammar.github.io/illustrated-transformer/

Self Attention & Transformers - Steps



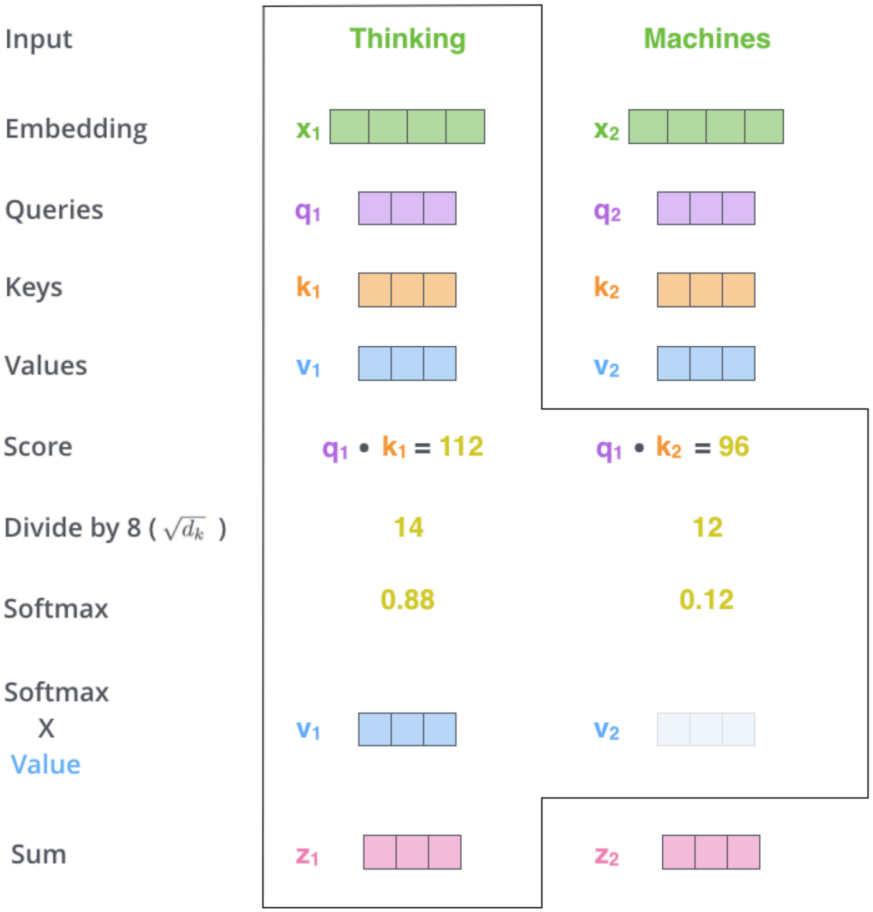
Ref : http://jalammar.github.io/illustrated-transformer/

Self Attention & Transformers - Steps



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

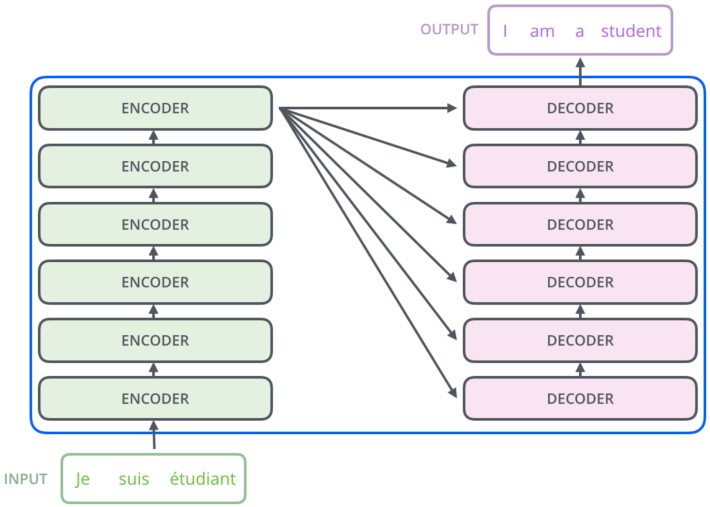
Self Attention & Transformers - Steps



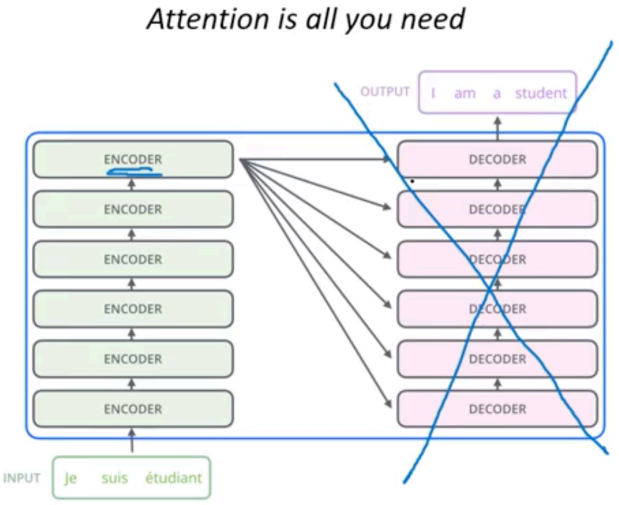
Ref : http://jalammar.github.io/illustrated-transformer/

Bidirectional Encoder Representations from Transformers

Transformer Architecture



BERT Architecture



Bidirectional Encoder Representations from Transformers

BERT's "Fake Tasks"

1. Masked Language Model
   - Predict the masked word.

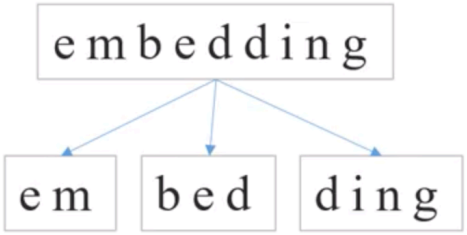
2. Next Sentence Prediction
   - Was sentence B found immediately after sentence A?

Bidirectional Encoder Representations from Transformers – Word Piece Tokenizer

BERT's Vocabulary

1. BERT is pre-trained → Vocabulary is fixed

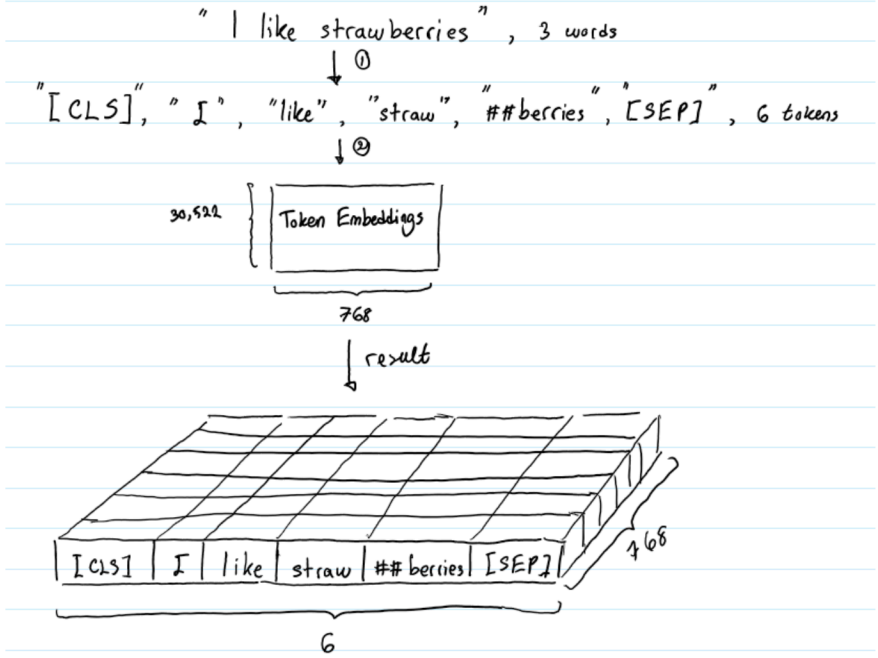
2. Break down **unknown words** into **subwords**:

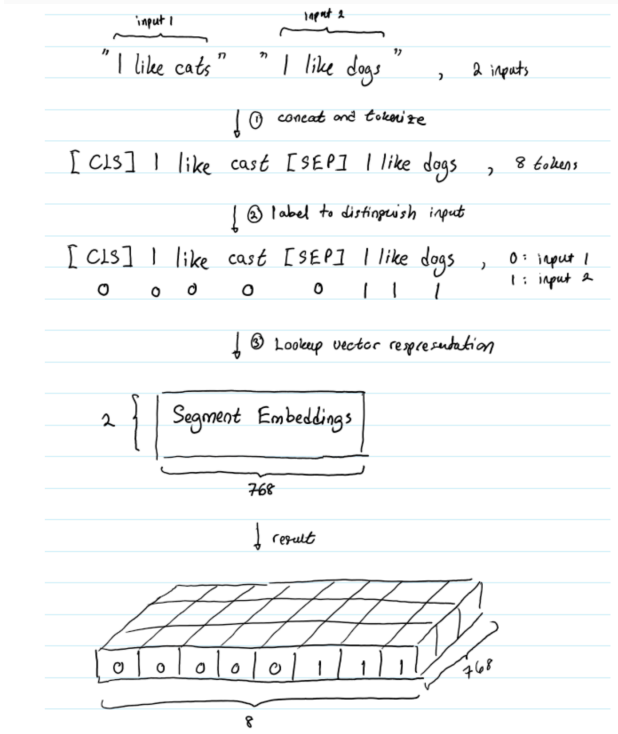


3. A subword exists for every character:

k r o x l d y p h i v c

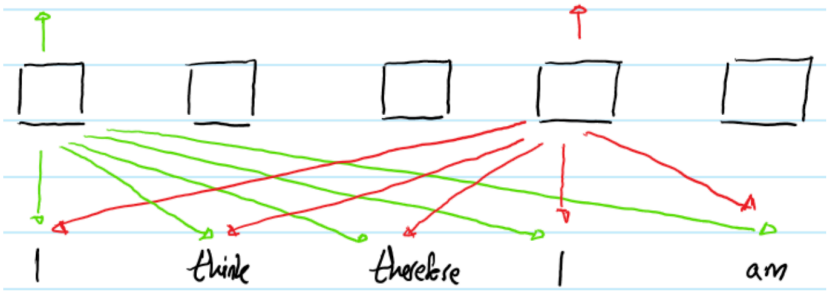
Bidirectional Encoder Representations from Transformers – Word Piece Tokenizer





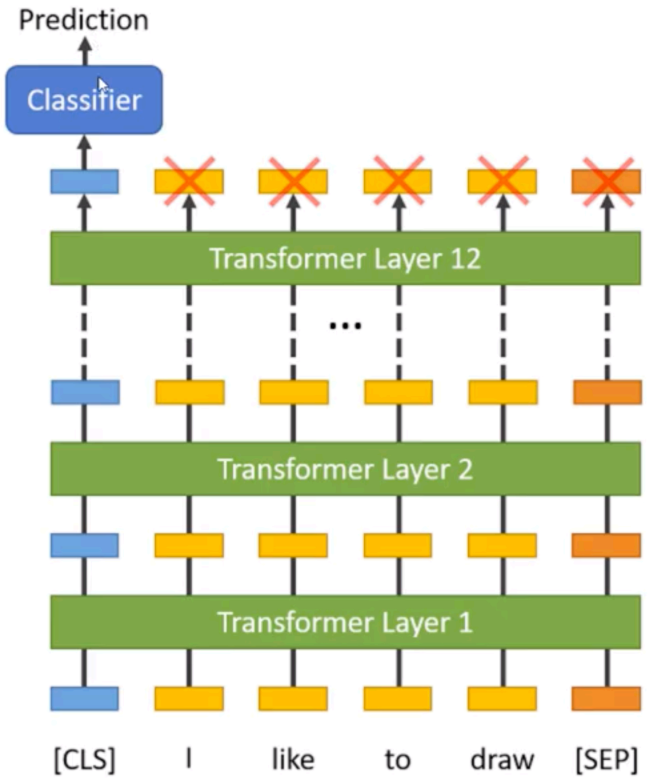
https://medium.com/@_init_/how-self-attention-with-relative-position-representations-works-28173b8c245a

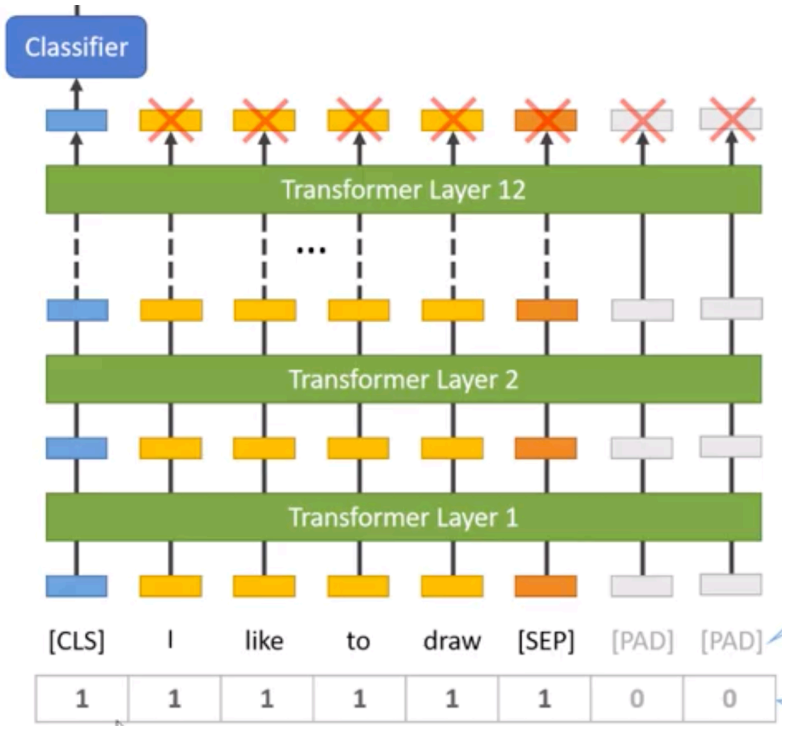
Bidirectional Encoder Representations from Transformers – Position Embedding



Index	Interpretation
0	dist between word at position i and i - 4
1	dist between word at position i and i - 3
2	dist between word at position i and i - 2
3	dist between word at position i and i - 1
4	dist between word at position i and i
5	dist between word at position i and i + 1
6	dist between word at position i and i + 2
7	dist between word at position i and i + 3
8	dist between word at position i and i + 4

Bidirectional Encoder Representations from Transformers – Tokens





Code walkthrough. !!!!

Thank you