LLMs SLMs and LMMs

The Power of Attention

Ninan Sajeeth Philip

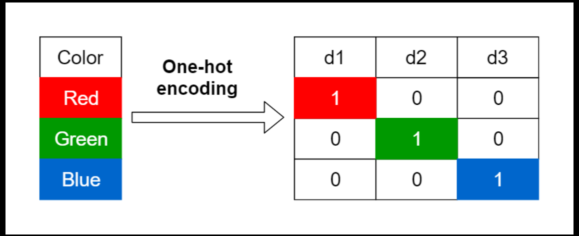
Artificial Intelligence Research and Intelligent Systems (airis4D)

December 28, 2024

Language as Data

- Corpus (Internet of Texts !!)
- Books
- Chapters
- Paragraphs
- Sentences
- Words
- Grammar

Enoding and Embedding



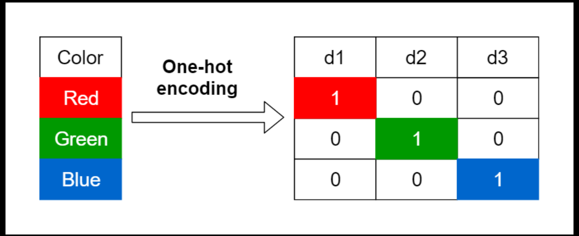
- 1. One Hot Encoding every category is represented by a new binary column.

[a]

[a] Source: [View link]

Computers need everything as numbers. Hence, categorical data (text or image) has to be converted to numeric values. Both encodings and Ebeddings do it, but there is a difference.

Enoding and Embedding



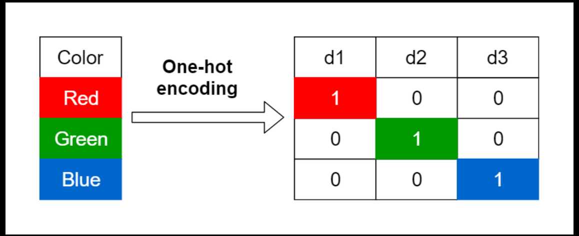
- 1. One Hot Encoding every category is represented by a new binary column.
  2. Simple and Interpretable but not scalable

[a]

[a] Source: [View link]

Computers need everything as numbers. Hence, categorical data (text or image) has to be converted to numeric values. Both encodings and Ebeddings do it, but there is a difference.

Enoding and Embedding



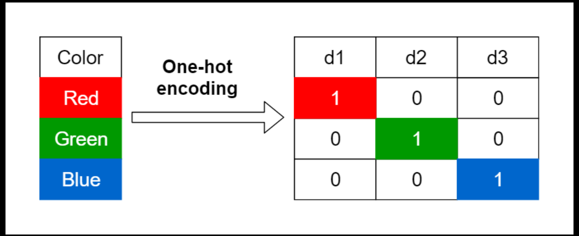
- 1. One Hot Encoding every category is represented by a new binary column.
  2. Simple and Interpretable but not scalable
  3. Bag of Words Words are ranked based on their frequency of occurrence

[a]

[a]Source: [View link]

Computers need everything as numbers. Hence, categorical data (text or image) has to be converted to numeric values. Both encodings and Ebeddings do it, but there is a difference.

Enoding and Embedding



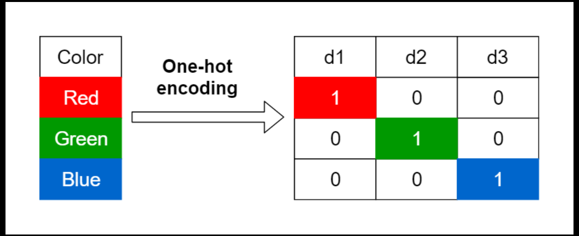
- 1. One Hot Encoding every category is represented by a new binary column.
  2. Simple and Interpretable but not scalable
  3. Bag of Words Words are ranked based on their frequency of occurrence
  4. Word Embeddings

[a]

[a]Source: [View link]

Computers need everything as numbers. Hence, categorical data (text or image) has to be converted to numeric values. Both encodings and Ebeddings do it, but there is a difference.

One-hot Encoding

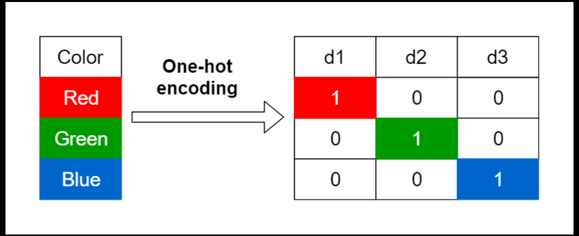


- 1. Treats each category as independent and orthogonal.

[a]

[a] Source: [View link]

One-hot Encoding

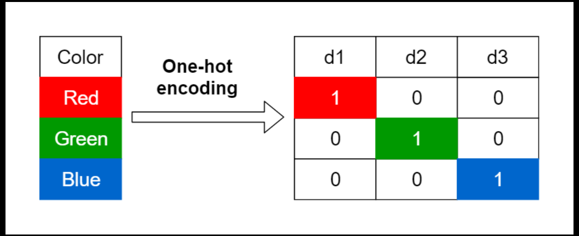


- 1. Treats each category as independent and orthogonal.
  2. Encoded vectors are easily interpretable.

[a]

[a]Source: [View link]

One-hot Encoding

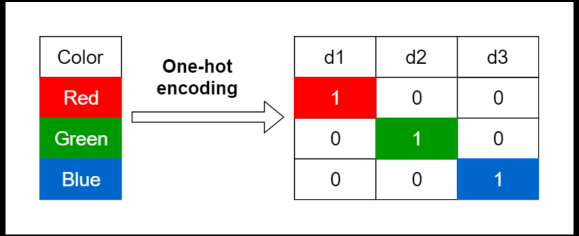


- 1. Treats each category as independent and orthogonal.
  2. Encoded vectors are easily interpretable.
  3. Does not require any learning.

[a]

[a] Source: [View link]

One-hot Encoding

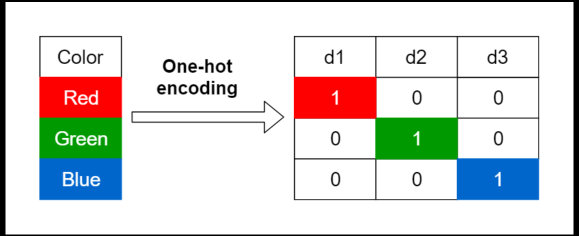


- 1. Treats each category as independent and orthogonal.
  2. Encoded vectors are easily interpretable.
  3. Does not require any learning.
  4. Increases the data dimensionality by creating a new binary column for each category.

[a]

[a] Source: [View link]

One-hot Encoding



- 1. Treats each category as independent and orthogonal.
  2. Encoded vectors are easily interpretable.
  3. Does not require any learning.
  4. Increases the data dimensionality by creating a new binary column for each category.
  5. Inefficient and sparse when dealing with large number of categorical features.

[a]

[a] Source: [View link]

Embedding

- 1. Embedding: reduces the dimensionality by representing each category as a dense vector of lower dimensionality (e.g., 8, 16, 32 dimensions).

Embedding

- 1. Embedding: reduces the dimensionality by representing each category as a dense vector of lower dimensionality (e.g., 8, 16, 32 dimensions).
  2. Embedding Captures semantic relationships and similarities between categories by placing similar categories closer together in the embedding space.

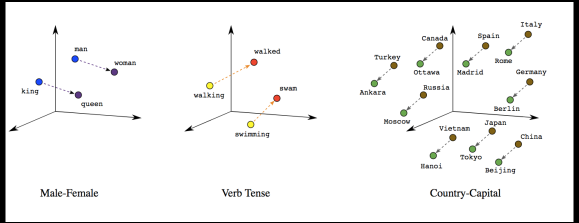
Embedding

- 1. Embedding: reduces the dimensionality by representing each category as a dense vector of lower dimensionality (e.g., 8, 16, 32 dimensions).
  2. Embedding Captures semantic relationships and similarities between categories by placing similar categories closer together in the embedding space.
  3. Embeddings are scalable and efficient for high-cardinality features.

Embedding

- 1. Embedding: reduces the dimensionality by representing each category as a dense vector of lower dimensionality (e.g., 8, 16, 32 dimensions).
  2. Embedding Captures semantic relationships and similarities between categories by placing similar categories closer together in the embedding space.
  3. Embeddings are scalable and efficient for high-cardinality features.
  4. Embeddings are adjusted during training to capture the relationships between categories, making them data-driven and context-aware.

Word Embeddings



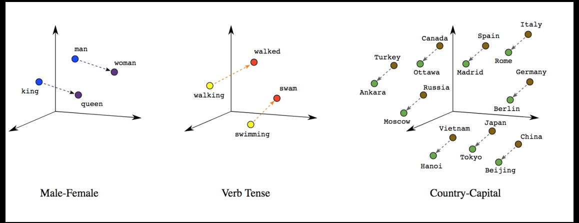
- 1. Word Embeddings encapsulates the word meaning in different contexts.

a

a Source: [View link]

1. How a person behaves in different situations can be given as scores on his personality traits as numbers. Word embeddings are like that. It is a set of numbers (vector or tensor) that encapsulates the contextual behaviour of the token so that it can be used to create the appropriate context vector based on a given context.

Word Embeddings



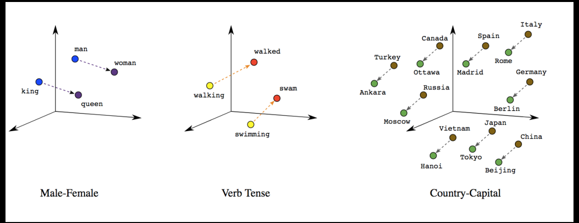
- 1. Word Embeddings encapsulates the word meaning in different contexts.
  2. A PCA on the Embeddings demonstrates how similar entities are clustered together in the embedded space.

a

[a] Source: [View link]

1. How a person behaves in different situations can be given as scores on his personality traits as numbers. Word embeddings are like that. It is a set of numbers (vector or tensor) that encapsulates the contextual behaviour of the token so that it can be used to create the appropriate context vector based on a given context.

Word Embeddings



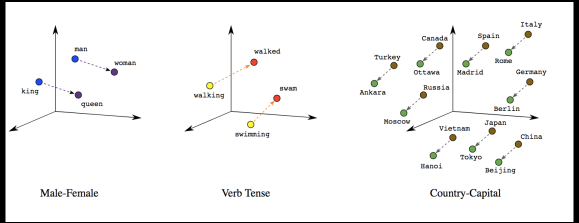
- 1. Word Embeddings encapsulates the word meaning in different contexts.
  2. A PCA on the Embeddings demonstrates how similar entities are clustered together in the embedded space.
  3. Higher dimension makes them non-interpretable but are scalable.

a

[a] Source: [View link]

1. How a person behaves in different situations can be given as scores on his personality traits as numbers. Word embeddings are like that. It is a set of numbers (vector or tensor) that encapsulates the contextual behaviour of the token so that it can be used to create the appropriate context vector based on a given context.

Word Embeddings



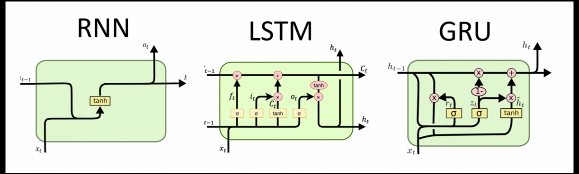
- 1. Word Embeddings encapsulates the word meaning in different contexts.
  2. A PCA on the Embeddings demonstrates how similar entities are clustered together in the embedded space.
  3. Higher dimension makes them non-interpretable but are scalable.
  4. Example: GloVe and Word2Vec

a

[a] Source: [View link]

1. How a person behaves in different situations can be given as scores on his personality traits as numbers. Word embeddings are like that. It is a set of numbers (vector or tensor) that encapsulates the contextual behaviour of the token so that it can be used to create the appropriate context vector based on a given context.

Pre Tranformer Models - RNN, LSTM and GRU



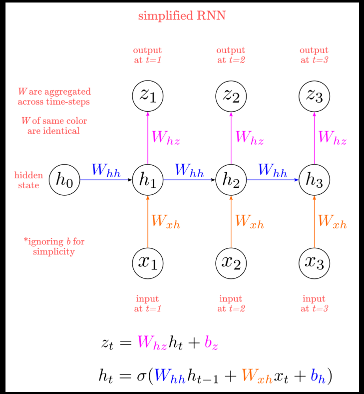
- Recurrent Neural Networks (RNN)
- Long Short-Term Memory (LSTM)
- Gated Recurrent Units (GRU)

[a]

[a] Source: [View link]

All of them use word embeddings on the input tokens.

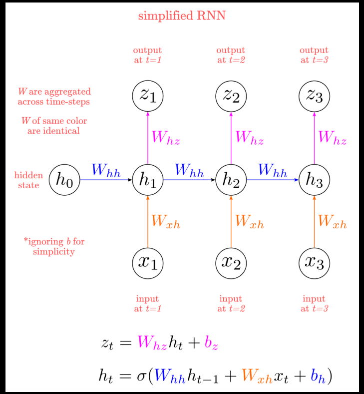
Recurrent Neural Network



1. $z_t = W_{hz} h_t + b_z$
2. $h_z = \sigma(W_{hh} h_{t-1} + W_{xh} x_t + b_h)$

1. At each time step, t, the RNN takes an input vector, x_t, and updates its hidden state, h_t, using the equation: $h_t = \sigma_k(W_{xh} x_i + W_{hh} h_i + b_k)$ where W_{xh} is the weight matrix between input and hidden layer, W_{hh} is the weight matrix for the recurrent connection, b_h is the bias vector and σ_k is the activation function (hyperbolic tanh function or RELU)
2. The calculation of gradients encounters terms involving the product of many Jacobian matrices. If the eigenvalues of J_k are less than 1, the product of these matrices will tend to zero as n increases, leading to vanishing gradients. Conversely, if the eigenvalues of J_k are greater than 1, the gradients can grow exponentially

Recurrent Neural Network



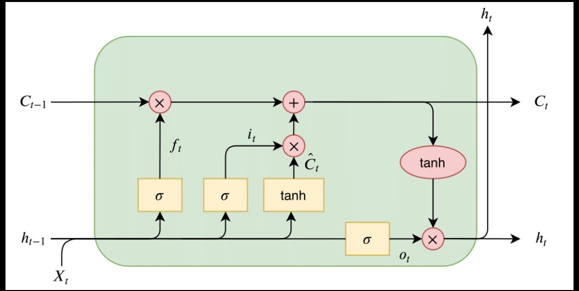
1. $z_t = W_{hz} h_t + b_z$
2. $h_z = \sigma(W_{hh} h_{t-1} + W_{xh} x_t + b_h)$
3. Issues: Vanishing and Exploding Gradients as the gradient of the error are backpropagated in time causing it to vanish if $\ll 1.0$ or explode if $\gg 1.0$

[a]

[a] Source: [View link]

1. At each time step, t, the RNN takes an input vector, x_t, and updates its hidden state, h_t, using the equation: $h_t = \sigma_k(W_{xh} x_i + W_{hh} h_i + b_k)$ where W_{xh} is the weight matrix between input and hidden layer, W_{hh} is the weight matrix for the recurrent connection, b_h is the bias vector and σ_k is the activation function (hyperbolic tanh function or RELU)
2. The calculation of gradients encounters terms involving the product of many Jacobian matrices. If the eigenvalues of J_k are less than 1, the product of these matrices will tend to zero as n increases, leading to vanishing gradients. Conversely, if the eigenvalues of J_k are greater than 1, the gradients can grow exponentially

Long Short-Term Memory Networks (LSTM)



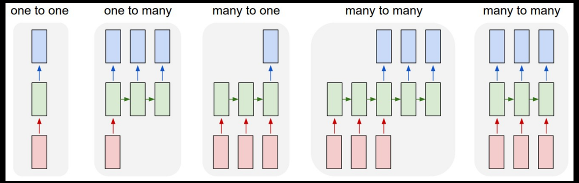
- LSTM use gating mechanisms (input, forget,output) to control the flow of information through the network over a longer period through the cell state C_t to prevent vanishing gradient problem.
- C_t transfers relevant information across different time steps.

[a]

[a] Source: [View link]

These gates determine how much of the input to consider, how much of the previous state to forget, and how much of the cell state to output.

Encoder-Decoder Architecture

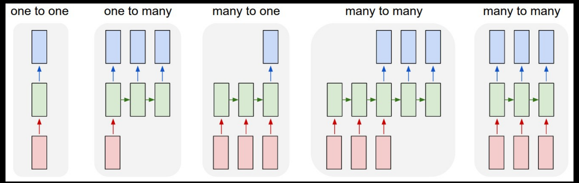


[a]

- What if the number of inputs differs from the number of outputs? For example, translation from one language to another?

[a]Source: [View link]

One2One to Many2Many Architectures

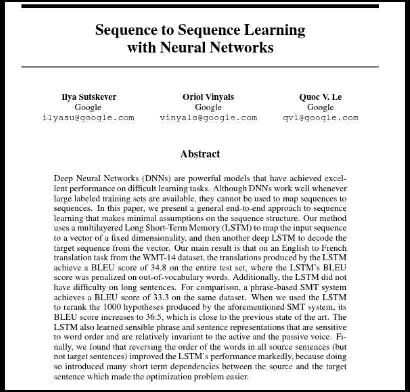


- One to One: Simple, vanilla model
- One to many: image to text conversion
- Many to One: Text to Image Generation
- Many to Many: Text translation

[a]

[a] Source: [View link]

Sequence to Sequence Paper (2014)



Sequence to Sequence Learning with Neural Networks

Ilya Sutskever Google ilyasu@google.com
Oriol Vinyals Google vinyals@google.com
Quoc V. Le Google qvl@google.com

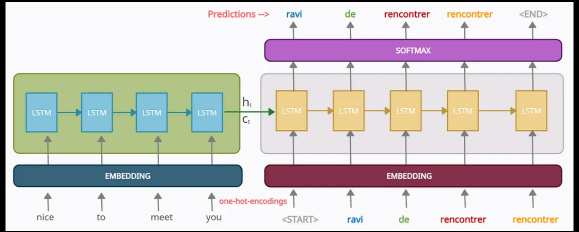
Abstract

Deep Neural Networks (DNNs) are powerful models that have achieved excellent performance on difficult learning tasks. Although DNNs work well whenever large labeled training sets are available, they cannot be used to map sequences to sequences. In this paper, we present a general end-to-end approach to sequence learning that makes minimal assumptions on the sequence structure. Our method uses a multilayered Long Short-Term Memory (LSTM) to map the input sequence to a vector of a fixed dimensionality, and then another deep LSTM to decode the target sequence from the vector. Our main result is that on an English to French translation task from the WMT-14 dataset, the translations produced by the LSTM achieve a BLEU score of 34.8 on the entire test set, where the LSTM's BLEU score was penalized on out-of-vocabulary words. Additionally, the LSTM did not have difficulty on long sentences. For comparison, a phrase-based SMT system achieves a BLEU score of 33.3 on the same dataset. When we used the LSTM to rerank the 1000 hypotheses produced by the aforementioned SMT system, its BLEU score increases to 36.5, which is close to the previous state of the art. The LSTM also learned sensible phrase and sentence representations that are sensitive to word order and are relatively invariant to the active and the passive voice. Finally, we found that reversing the order of the words in all source sentences (but not target sentences) improved the LSTM's performance markedly, because doing so introduced many short term dependencies between the source and the target sentence which made the optimization problem easier.

- Encoder: The Encoder processes each token in the input sequence to construct the fixed-length context vector.
- Context vector: A vector encoded with all the information in the input sequence.
- Decoder: Converts context vector to predict the target-sequence token by token.

For further information on how this encoder-decoder architecture works in seq2seq learning, Please refer to this paper.

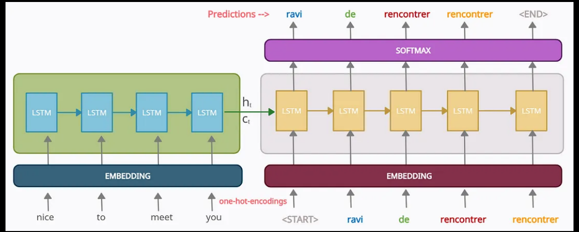
Sequence to Sequence



1. LSTM generates fixed length context vector.

Challenge: The context vector should be able to hold the complete information in the input sequence - which is a challenge if the information content is large.

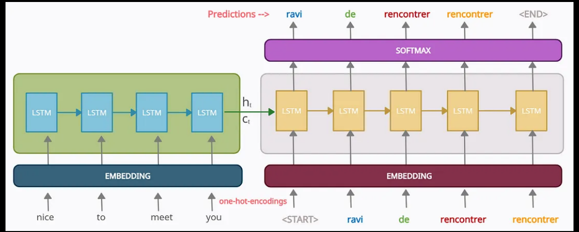
Sequence to Sequence



1. LSTM generates fixed length context vector.
2. The Decoder predicts a set of tokens that goes to a softmax function to predict the most probable token.

Challenge: The context vector should be able to hold the complete information in the input sequence - which is a challenge if the information content is large.

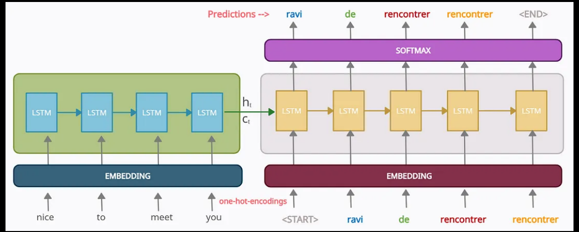
Sequence to Sequence



1. LSTM generates fixed length context vector.
2. The Decoder predicts a set of tokens that goes to a softmax function to predict the most probable token.
3. The most probable token found is now applied to the context vector to find the next token.

Challenge: The context vector should be able to hold the complete information in the input sequence - which is a challenge if the information content is large.

Sequence to Sequence



1. LSTM generates fixed length context vector.
2. The Decoder predicts a set of tokens that goes to a softmax function to predict the most probable token.
3. The most probable token found is now applied to the context vector to find the next token.
4. Challenge: The fixed length context vector should have the complete information in the input sequence - What if the information content is large?

[a]

[a] Source: [View link]

Challenge: The context vector should be able to hold the complete information in the input sequence - which is a challenge if the information content is large.

Sequence to Sequence - Captures Context

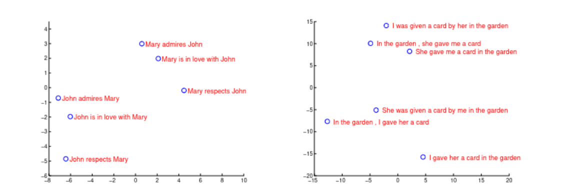


Figure 2: The figure shows a 2-dimensional PCA projection of the LSTM hidden states that are obtained after processing the phrases in the figures. The phrases are clustered by meaning, which in these examples is primarily a function of word order, which would be difficult to capture with a bag-of-words model. Notice that both clusters have similar internal structure.

1. The most significant feature of seq2seq learning is that it can capture the context efficiently and cluster context vectors in terms of their meaning.

The most significant feature of seq2seq learning is that it could capture the context efficiently and cluster context vectors in terms of their meaning unheard of in one-hot encoding or bag of wordsunlike in one-hot encoding or bag of words where there is no link between the code and the content it represents, the embedding layer, in this case, encapsulates the contextual behaviour of the word.To give an example for how word2Vec works, it is like the scores one may have describing their behaviour in different circumstances.

Sequence to Sequence - Captures Context

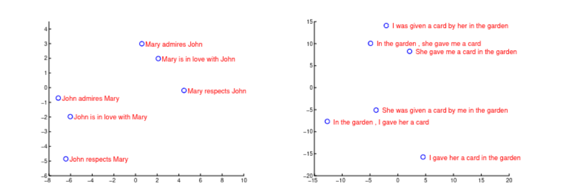
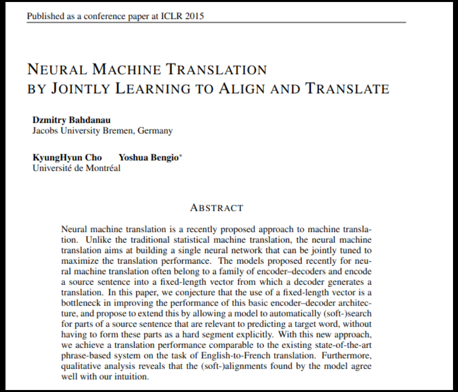


Figure 2: The figure shows a 2-dimensional PCA projection of the LSTM hidden states that are obtained after processing the phrases in the figures. The phrases are clustered by meaning, which in these examples is primarily a function of word order, which would be difficult to capture with a bag-of-words model. Notice that both clusters have similar internal structure.

1. The most significant feature of seq2seq learning is that it can capture the context efficiently and cluster context vectors in terms of their meaning.
2. Word Embeddings cluster words depending on their possible contextual meanings.
3. Seq2Seq Learning cluster sequences based on their information content.

The most significant feature of seq2seq learning is that it could capture the context efficiently and cluster context vectors in terms of their meaning unheard of in one-hot encoding or bag of wordsunlike in one-hot encoding or bag of words where there is no link between the code and the content it represents, the embedding layer, in this case, encapsulates the contextual behaviour of the word.To give an example for how word2Vec works, it is like the scores one may have describing their behaviour in different circumstances.

Attention Mechanism (2015)


Published as a conference paper at ICLR 2015

NEURAL MACHINE TRANSLATION
BY JOINTLY LEARNING TO ALIGN AND TRANSLATE

Dzmitry Bahdanau
Jacobs University Bremen, Germany

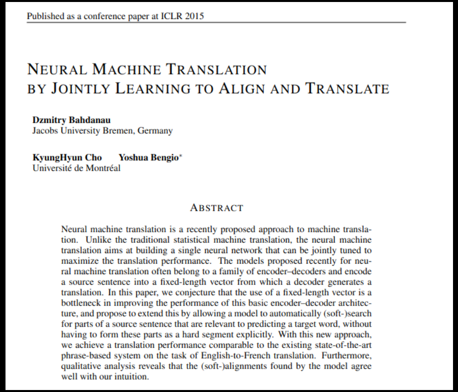
KyungHyun Cho **Yoshua Bengio**[*]
Université de Montréal

ABSTRACT

Neural machine translation is a recently proposed approach to machine translation. Unlike the traditional statistical machine translation, the neural machine translation aims at building a single neural network that can be jointly tuned to maximize the translation performance. The models proposed recently for neural machine translation often belong to a family of encoder–decoders and encode a source sentence into a fixed-length vector from which a decoder generates a translation. In this paper, we conjecture that the use of a fixed-length vector is a bottleneck in improving the performance of this basic encoder–decoder architecture, and propose to extend this by allowing a model to automatically (soft-)search for parts of a source sentence that are relevant to predicting a target word, without having to form these parts as a hard segment explicitly. With this new approach, we achieve a translation performance comparable to the existing state-of-the-art phrase-based system on the task of English-to-French translation. Furthermore, qualitative analysis reveals that the (soft-)alignments found by the model agree well with our intuition.

1. Introduced the first attention mechanism for neural machine translation

Attention Mechanism (2015)


Published as a conference paper at ICLR 2015

NEURAL MACHINE TRANSLATION BY JOINTLY LEARNING TO ALIGN AND TRANSLATE

Dzmitry Bahdanau
Jacobs University Bremen, Germany

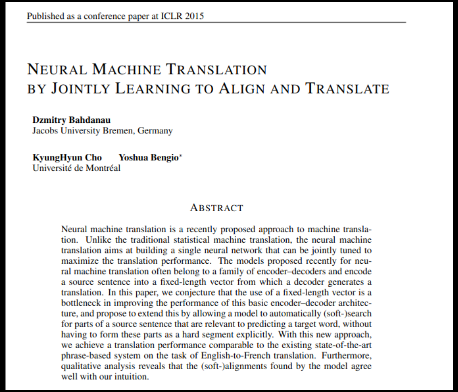
KyungHyun Cho Yoshua Bengio*
Université de Montréal

ABSTRACT

Neural machine translation is a recently proposed approach to machine translation. Unlike the traditional statistical machine translation, the neural machine translation aims at building a single neural network that can be jointly tuned to maximize the translation performance. The models proposed recently for neural machine translation often belong to a family of encoder–decoders and encode a source sentence into a fixed-length vector from which a decoder generates a translation. In this paper, we conjecture that the use of a fixed-length vector is a bottleneck in improving the performance of this basic encoder–decoder architecture, and propose to extend this by allowing a model to automatically (soft-)search for parts of a source sentence that are relevant to predicting a target word, without having to form these parts as a hard segment explicitly. With this new approach, we achieve a translation performance comparable to the existing state-of-the-art phrase-based system on the task of English-to-French translation. Furthermore, qualitative analysis reveals that the (soft-)alignments found by the model agree well with our intuition.

1. Introduced the first attention mechanism for neural machine translation
2. No need to encode all the information in a sentence and its context into a single vector.

Attention Mechanism (2015)



Published as a conference paper at ICLR 2015

NEURAL MACHINE TRANSLATION
BY JOINTLY LEARNING TO ALIGN AND TRANSLATE

Dzmitry Bahdanau
Jacobs University Bremen, Germany

KyungHyun Cho **Yoshua Bengio***
Université de Montréal

ABSTRACT

Neural machine translation is a recently proposed approach to machine translation. Unlike the traditional statistical machine translation, the neural machine translation aims at building a single neural network that can be jointly tuned to maximize the translation performance. The models proposed recently for neural machine translation often belong to a family of encoder–decoders and encode a source sentence into a fixed-length vector from which a decoder generates a translation. In this paper, we conjecture that the use of a fixed-length vector is a bottleneck in improving the performance of this basic encoder–decoder architecture, and propose to extend this by allowing a model to automatically (soft-)search for parts of a source sentence that are relevant to predicting a target word, without having to form these parts as a hard segment explicitly. With this new approach, we achieve a translation performance comparable to the existing state-of-the-art phrase-based system on the task of English-to-French translation. Furthermore, qualitative analysis reveals that the (soft-)alignments found by the model agree well with our intuition.

1. Introduced the first attention mechanism for neural machine translation
2. No need to encode all the information in a sentence and its context into a single vector.
3. The model automatically search for relevant parts of a sentence for predicting a target word, instead of explicitly depending on a single Context Vector.

Attention Mechanism Key Concepts

3 LEARNING TO ALIGN AND TRANSLATE

In this section, we propose a novel architecture for neural machine translation. The new architecture consists of a bidirectional RNN as an encoder (Sec. 3.2) and a decoder that emulates searching through a source sentence during decoding a translation (Sec. 3.1).

3.1 DECODER: GENERAL DESCRIPTION

In a new model architecture, we define each conditional probability in Eq. (2) as:

$$p(y_i|y_1, \ldots, y_{i-1}, \mathbf{x}) = g(y_{i-1}, s_i, c_i), \quad (4)$$

where s_i is an RNN hidden state for time i, computed by

$$s_i = f(s_{i-1}, y_{i-1}, c_i).$$

It should be noted that unlike the existing encoder–decoder approach (see Eq. (2)), here the probability is conditioned on a distinct context vector c_i for each target word y_i.

The context vector c_i depends on a sequence of *annotations* $(h_1, \cdots, h_{T_x})$ to which an encoder maps the input sentence. Each annotation h_i contains information about the whole input sequence with a strong focus on the parts surrounding the i-th word of the input sequence. We explain in detail how the annotations are computed in the next section.

The context vector c_i is, then, computed as a weighted sum of these annotations h_i:

$$c_i = \sum^{T_x} \alpha_{ij} h_j. \quad (5)$$

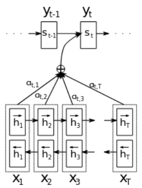
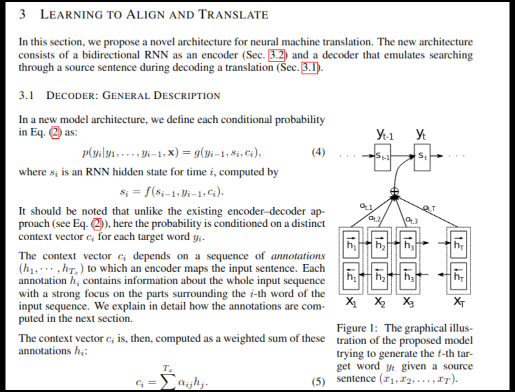


Figure 1: The graphical illustration of the proposed model trying to generate the t-th target word y_t given a source sentence $(x_1, x_2, \ldots, x_T)$.

1. Each annotation is created by concatenating forward and backward bidirectional RNN states.

Attention Mechanism Key Concepts



3 LEARNING TO ALIGN AND TRANSLATE

In this section, we propose a novel architecture for neural machine translation. The new architecture consists of a bidirectional RNN as an encoder (Sec. 3.2) and a decoder that emulates searching through a source sentence during decoding a translation (Sec. 3.1).

3.1 DECODER: GENERAL DESCRIPTION

In a new model architecture, we define each conditional probability in Eq. (2) as:

$$p(y_i|y_1, \ldots, y_{i-1}, \mathbf{x}) = g(y_{i-1}, s_i, c_i), \quad (4)$$

where s_i is an RNN hidden state for time i, computed by

$$s_i = f(s_{i-1}, y_{i-1}, c_i).$$

It should be noted that unlike the existing encoder–decoder approach (see Eq. (2)), here the probability is conditioned on a distinct context vector c_i for each target word y_i.

The context vector c_i depends on a sequence of *annotations* $(h_1, \cdots, h_{T_x})$ to which an encoder maps the input sentence. Each annotation h_i contains information about the whole input sequence with a strong focus on the parts surrounding the i-th word of the input sequence. We explain in detail how the annotations are computed in the next section.

The context vector c_i is, then, computed as a weighted sum of these annotations h_i:

$$c_i = \sum^{T_x} \alpha_{ij} h_j. \quad (5)$$

Figure 1: The graphical illustration of the proposed model trying to generate the t-th target word y_t given a source sentence $(x_1, x_2, \ldots, x_T)$.

1. Each annotation is created by concatenating forward and backward bidirectional RNN states.
2. h_i contains information about the whole input sequence with a strong focus on the parts surrounding the i^{th} word of the sequence.

Attention Mechanism Key Concepts

3 LEARNING TO ALIGN AND TRANSLATE

In this section, we propose a novel architecture for neural machine translation. The new architecture consists of a bidirectional RNN as an encoder (Sec. 3.2) and a decoder that emulates searching through a source sentence during decoding a translation (Sec. 3.1).

3.1 DECODER: GENERAL DESCRIPTION

In a new model architecture, we define each conditional probability in Eq. (2) as:

$$p(y_i|y_1, \ldots, y_{i-1}, \mathbf{x}) = g(y_{i-1}, s_i, c_i), \quad (4)$$

where s_i is an RNN hidden state for time i, computed by

$$s_i = f(s_{i-1}, y_{i-1}, c_i).$$

It should be noted that unlike the existing encoder–decoder approach (see Eq. (2)), here the probability is conditioned on a distinct context vector c_i for each target word y_i.

The context vector c_i depends on a sequence of *annotations* $(h_1, \cdots, h_{T_x})$ to which an encoder maps the input sentence. Each annotation h_i contains information about the whole input sequence with a strong focus on the parts surrounding the i-th word of the input sequence. We explain in detail how the annotations are computed in the next section.

The context vector c_i is, then, computed as a weighted sum of these annotations h_i:

$$c_i = \sum^{T_x} \alpha_{ij} h_j. \quad (5)$$

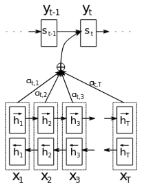
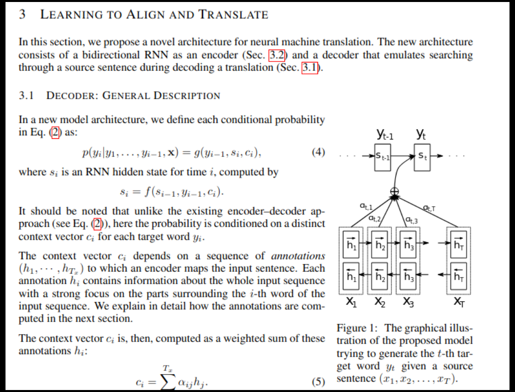


Figure 1: The graphical illustration of the proposed model trying to generate the t-th target word y_t given a source sentence $(x_1, x_2, \ldots, x_T)$.

1. Each annotation is created by concatenating forward and backward bidirectional RNN states.
2. h_i contains information about the whole input sequence with a strong focus on the parts surrounding the i^{th} word of the sequence.
3. The model dynamically updates the context vector C_i for each target word using a weighted sum of annotations.

Attention Mechanism Key Concepts



3 Learning to Align and Translate

In this section, we propose a novel architecture for neural machine translation. The new architecture consists of a bidirectional RNN as an encoder (Sec. 3.2) and a decoder that emulates searching through a source sentence during decoding a translation (Sec. 3.1).

3.1 Decoder: General Description

In a new model architecture, we define each conditional probability in Eq. (2) as:

$$p(y_i|y_1, \ldots, y_{i-1}, \mathbf{x}) = g(y_{i-1}, s_i, c_i), \quad (4)$$

where s_i is an RNN hidden state for time i, computed by

$$s_i = f(s_{i-1}, y_{i-1}, c_i).$$

It should be noted that unlike the existing encoder–decoder approach (see Eq. (2)), here the probability is conditioned on a distinct context vector c_i for each target word y_i.

The context vector c_i depends on a sequence of *annotations* $(h_1, \cdots, h_{T_x})$ to which an encoder maps the input sentence. Each annotation h_i contains information about the whole input sequence with a strong focus on the parts surrounding the i-th word of the input sequence. We explain in detail how the annotations are computed in the next section.

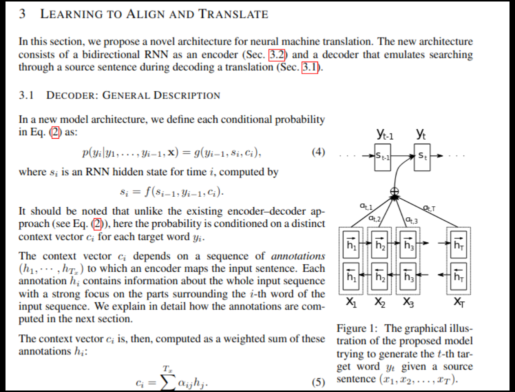
The context vector c_i is, then, computed as a weighted sum of these annotations h_i:

$$c_i = \sum^{T_x} \alpha_{ij} h_j. \quad (5)$$

Figure 1: The graphical illustration of the proposed model trying to generate the t-th target word y_t given a source sentence $(x_1, x_2, \ldots, x_T)$.

1. Each annotation is created by concatenating forward and backward bidirectional RNN states.
2. h_i contains information about the whole input sequence with a strong focus on the parts surrounding the i^{th} word of the sequence.
3. The model dynamically updates the context vector C_i for each target word using a weighted sum of annotations.
4. The training for both models is done simulataniously using backpropagation.

Attention Mechanism Challenges



3 Learning to Align and Translate

In this section, we propose a novel architecture for neural machine translation. The new architecture consists of a bidirectional RNN as an encoder (Sec. 3.2) and a decoder that emulates searching through a source sentence during decoding a translation (Sec. 3.1).

3.1 Decoder: General Description

In a new model architecture, we define each conditional probability in Eq. (2) as:

$$p(y_i|y_1, \ldots, y_{i-1}, \mathbf{x}) = g(y_{i-1}, s_i, c_i), \quad (4)$$

where s_i is an RNN hidden state for time i, computed by

$$s_i = f(s_{i-1}, y_{i-1}, c_i).$$

It should be noted that unlike the existing encoder–decoder approach (see Eq. (2)), here the probability is conditioned on a distinct context vector c_i for each target word y_i.

The context vector c_i depends on a sequence of *annotations* $(h_1, \cdots, h_{T_x})$ to which an encoder maps the input sentence. Each annotation h_i contains information about the whole input sequence with a strong focus on the parts surrounding the i-th word of the input sequence. We explain in detail how the annotations are computed in the next section.

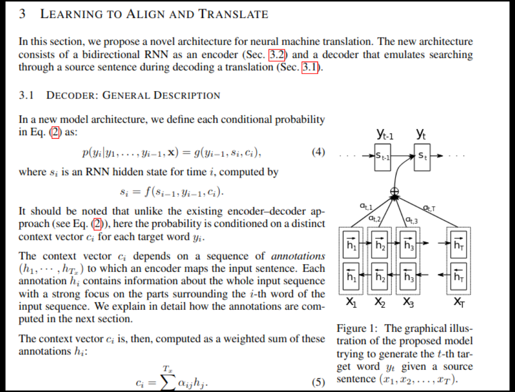
The context vector c_i is, then, computed as a weighted sum of these annotations h_i:

$$c_i = \sum^{T_x} \alpha_{ij} h_j. \quad (5)$$

Figure 1: The graphical illustration of the proposed model trying to generate the t-th target word y_t given a source sentence $(x_1, x_2, \ldots, x_T)$.

- Attention mechanism has efficiently handled the problem with long sequences and the exploding/vanishing gradient problems.

Attention Mechanism Challenges



3 Learning to Align and Translate

In this section, we propose a novel architecture for neural machine translation. The new architecture consists of a bidirectional RNN as an encoder (Sec. 3.2) and a decoder that emulates searching through a source sentence during decoding a translation (Sec. 3.1).

3.1 Decoder: General Description

In a new model architecture, we define each conditional probability in Eq. (2) as:

$$p(y_i|y_1, \ldots, y_{i-1}, \mathbf{x}) = g(y_{i-1}, s_i, c_i), \tag{4}$$

where s_i is an RNN hidden state for time i, computed by

$$s_i = f(s_{i-1}, y_{i-1}, c_i).$$

It should be noted that unlike the existing encoder–decoder approach (see Eq. (2)), here the probability is conditioned on a distinct context vector c_i for each target word y_i.

The context vector c_i depends on a sequence of *annotations* $(h_1, \cdots, h_{T_x})$ to which an encoder maps the input sentence. Each annotation h_i contains information about the whole input sequence with a strong focus on the parts surrounding the i-th word of the input sequence. We explain in detail how the annotations are computed in the next section.

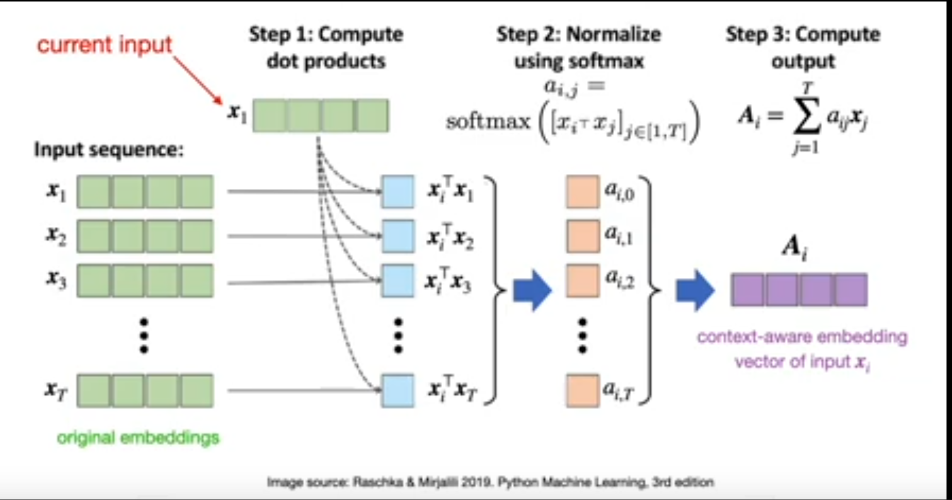
The context vector c_i is, then, computed as a weighted sum of these annotations h_i:

$$c_i = \sum^{T_x} \alpha_{ij} h_j. \tag{5}$$

Figure 1: The graphical illustration of the proposed model trying to generate the t-th target word y_t given a source sentence $(x_1, x_2, \ldots, x_T)$.

- Attention mechanism has efficiently handled the problem with long sequences and the exploding/vanishing gradient problems.
- Bottleneck Although the long sequence problem is now addressed by the attention mechanism, still the sequence is submitted with time stamps $X_1, X_2 \ldots X_t$, which means sequentially (one after another). This means the model is not scalable to be trained on large amounts of data.

Attention Recap



Also, the input word embedded vectors are fixed for each word, which means, there is no scope for any new learning and the output would be purely dependent on what was learned during the creation of the embedded vectors. But in LLMs, we want the machine to learn new contexts and meanings as it comes across huge volumes of new data.

From Sequential to Parallel Processing (2017)

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

- Attention is all you need
- BERT, GPT

Get rid of LSTM or RNN and use Self-attention that can handle all words at once with the positional encoding mechanism.It also uses Contextual Embeddings

Transformer AI Revolution

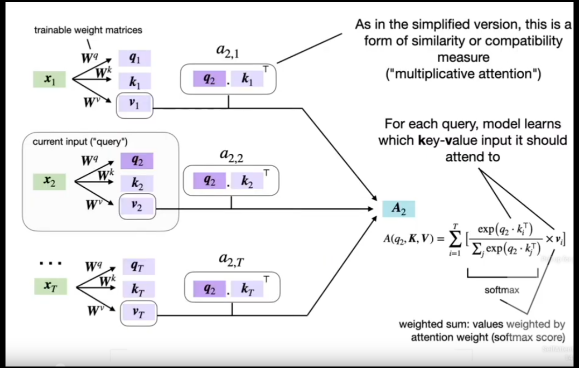
Attention Is All You Need

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

- Transformer is a Deep Learning Architecture using Attention Mechanism to handle sequential data in parallel and pay Attention to the connecting dots in the content and context it captures. (*personal definition*)

1. Over 70 % of the literature on AI nowadays uses a transformer architecture.
2. The name Transformer was coined by Jacob Iskariot, the 4th author in the paper by the Google team, who voted against the earlier name Attention Net as a catchy name.
3. In contrast to earlier models like RNN or LSTM or similar sequential learning models that had several disadvantages like exploding gradients and sequence lengths etc.,, Transformers, through its ability to process in parallel, is scalable to any amount of data making the revolution in AI possible.

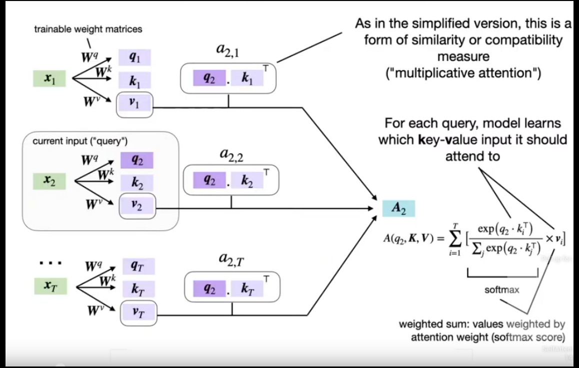
Self Attention



- *Query*, $q = W_q.x_i$
- *Key*, $k = W_k.x_i$
- *Value*, $v = W_v.x_i$[a]

The most important point here is that (a) The network can now learn new contexts, (b) The whole process can now be done in parallel,

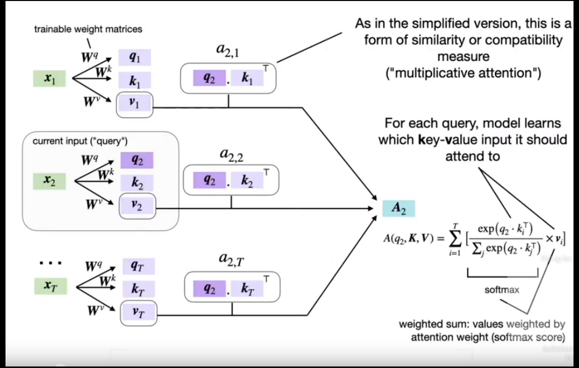
Self Attention



- $Query, q = W_q.x_i$
- $Key, k = W_k.x_i$
- $Value, v = W_v.x_i$[a]
-
- x is word embedding of dimension [1xN]
- W is Weight Matrix of dimension [NxM]
- q,k,v are thus having dimension [1xM]

The most important point here is that (a) The network can now learn new contexts, (b) The whole process can now be done in parallel,

Self Attention

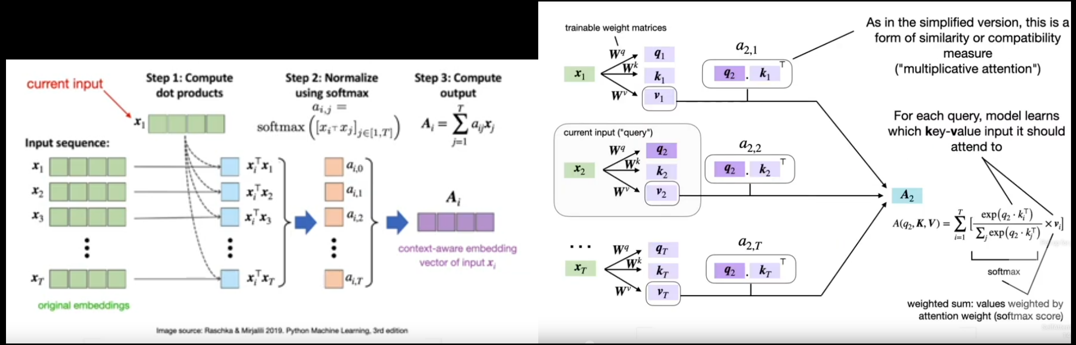


- $Query, \color{red}{q} = W_q.x_i$
- $Key, \color{red}{k} = W_k.x_i$
- $Value, \color{red}{v} = W_v.x_i$[a]
-
- x is word embedding of dimension [1xN]
- W is Weight Matrix of dimension [NxM]
- q,k,v are thus having dimension [1xM]
- $A(q_j, K, V) = \sum_{i=1}^{T} \frac{e^{q_j \cdot k_i^T}}{\sum_j e^{q_j \cdot k_j^T}} v_i$
- dot product gives a scalar and multiplying by the Value vector v gives us a [1xM] dimension vector

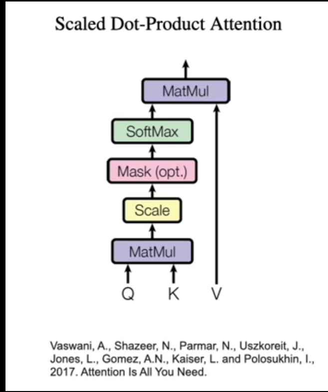
[a] Video Source

The most important point here is that (a) The network can now learn new contexts, (b) The whole process can now be done in parallel,

Attention vs Self Attention

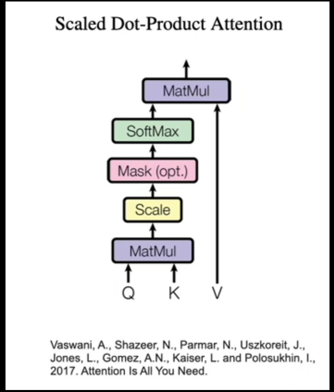


Self Attention



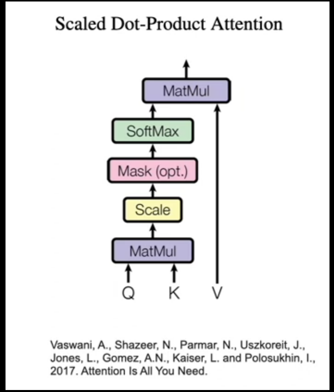
- Since the computation can be done in parallel, the Attention for the whole A(Q,K,V) can be computed by treating Q as the vector of dimension L x N where L is the sequence Length. $(Q, K, V) \rightarrow \mathbb{R}^{LxM}$

Self Attention



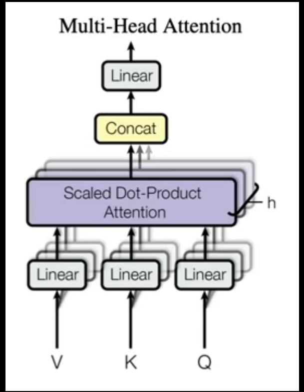
- Since the computation can be done in parallel, the Attention for the whole A(Q,K,V) can be computed by treating Q as the vector of dimension L x N where L is the sequence Length. $(Q, K, V) \to \mathbb{R}^{LxM}$
- Reulting A will be a matrix of dimension LxL and to prevent it from growing too large than the gradients, we scale it down by $\sqrt{M}$.

Self Attention

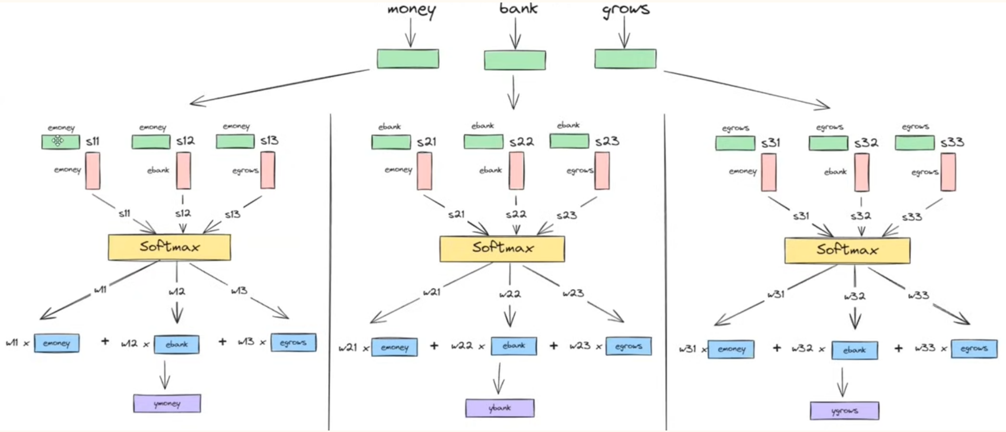


- Since the computation can be done in parallel, the Attention for the whole A(Q,K,V) can be computed by treating Q as the vector of dimension L x N where L is the sequence Length. $(Q, K, V) \to \mathbb{R}^{LxM}$
- Reulting A will be a matrix of dimension LxL and to prevent it from growing too large than the gradients, we scale it down by $\sqrt{M}$.
- Thus A(Q,K,V) $= softmax(\frac{QK^T}{\sqrt{M}})V$

Multi-head Attention

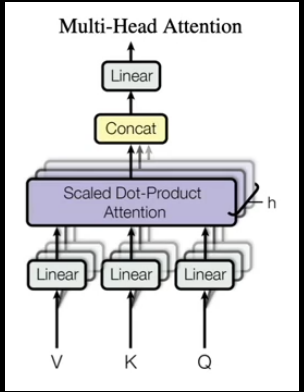


- Just like multiple channels use different kernels in CNNs to capture different details of the image, Multiple self-attentions with different weight matrices capture different information from the sequence!

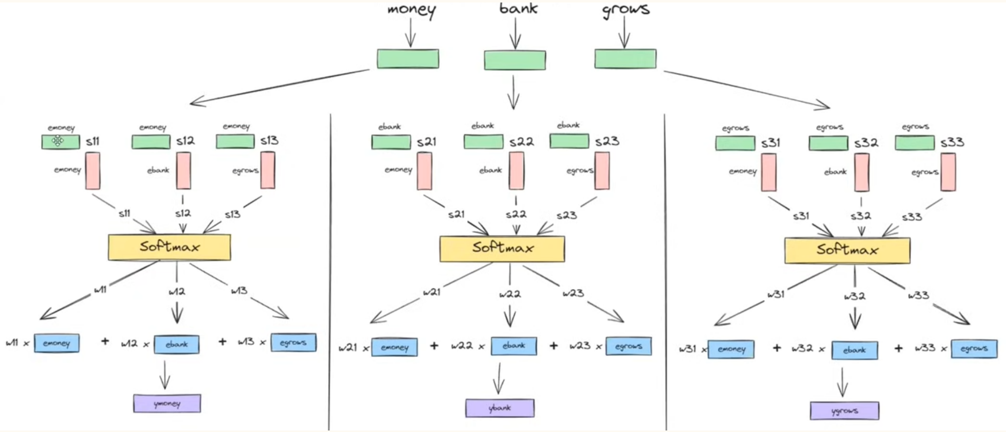


[Watch Video]: https://www.youtube.com/watch?v=-tCKPI$_{8}Xb$8

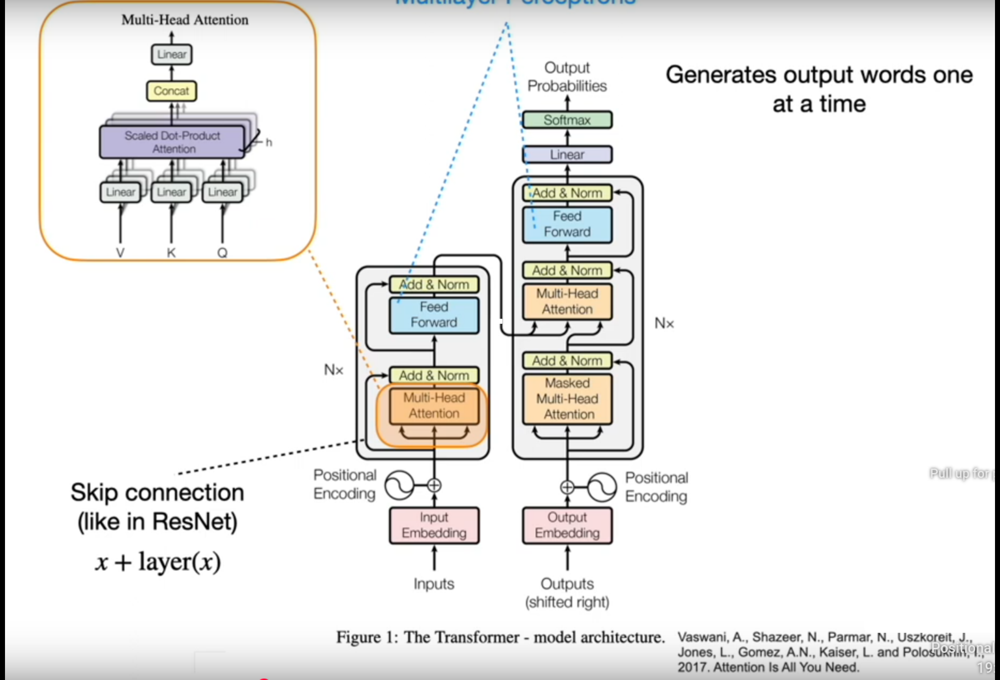
Multi-head Attention



- Just like multiple channels use different kernels in CNNs to capture different details of the image, Multiple self-attentions with different weight matrices capture different information from the sequence!
- To have the same dimension for the output from the multi head Attention as that of the self-attention, the dimension of the Weight matrix is kept as $\frac{M}{h}$ where h is the number of heads and concatenation will ensure that the resulting dimension is LxM.



[Watch Video]: https://www.youtube.com/watch?v=-tCKPI$_8Xb$8



1. The left is the Encoder, and the Right is the Decoder.
2. Generates one word at a time. (input English, output German)
3. skip connections
4. Masked Multihead attention - mask elements used for training- achieved by setting softmax values for them to $-\infty$
5. Softmax at output gives probabilities for the words in the dictionary
6. Positional Encoding - Sinusoidal positional encoding (skipping)

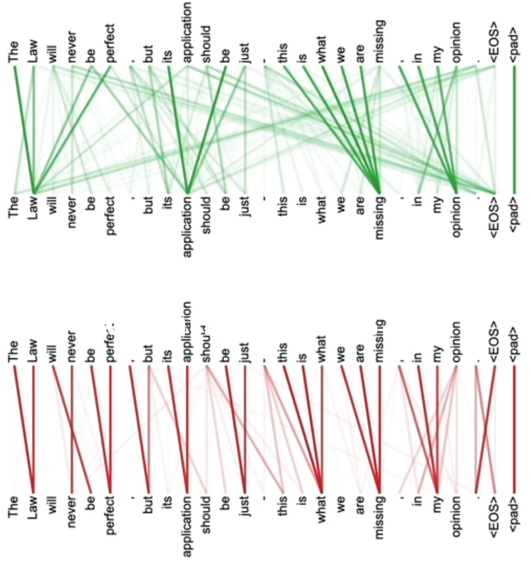
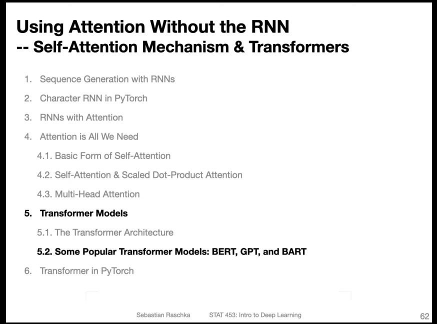


Figure 5: Many of the attention heads exhibit behaviour that seems related to the structure of the sentence. We give two such examples above, from two different heads from the encoder self-attention at layer 5 of 6. The heads clearly learned to perform different tasks.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L. and Polosukhin, I., 2017. Attention Is All You Need.

Self Supervised Learning



Click to play
Play

- Taking a corpus and creating labels by itself by masking or structure analysis of the sequence. (Q&A)

[a]

---

[a] YouTube video: [Watch Video]