LLMs SLMs and LMMs

The Power of Attention

Ninan Sajeeth Philip

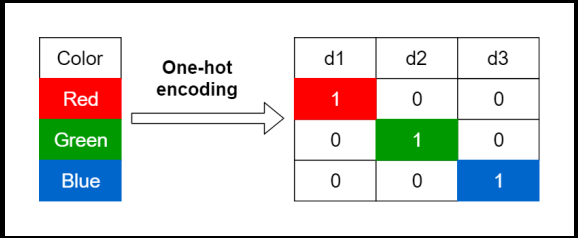
Artificial Intelligence Research and Intelligent Systems (airis4D)

January 6, 2025

Language as Data

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

One-hot Encoding

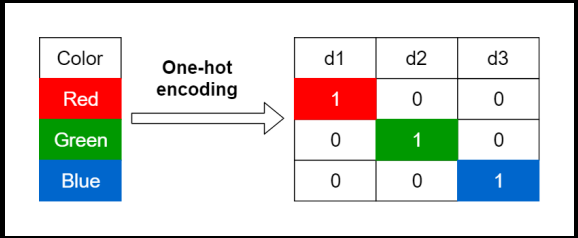


1. It uses an array of binary columns to represent each entity by a unique bit set to 1 while all others are 0.
2. Treats each entity as independent and orthogonal.
3. Encoded vectors are easily interpretable.
4. Does not require any learning.
5. Increases the data dimensionality by creating a new binary column for each category.
6. Inefficient and sparse when dealing with large number of categorical features.

[a]

[a] Source: [View link]

One-hot Encoding

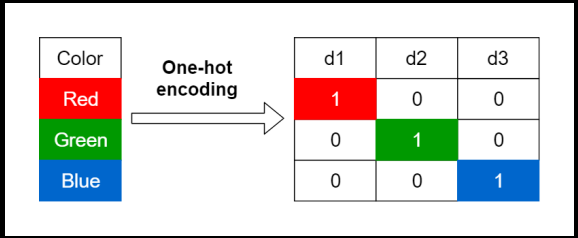


- 1. It uses an array of binary columns to represent each entity by a unique bit set to 1 while all others are 0.
- 2. Treats each entity as independent and orthogonal.
- 3. Encoded vectors are easily interpretable.
- 4. Does not require any learning.
- 5. Increases the data dimensionality by creating a new binary column for each category.
- 6. Inefficient and sparse when dealing with large number of categorical features.

[a]

[a] Source: [View link]

One-hot Encoding

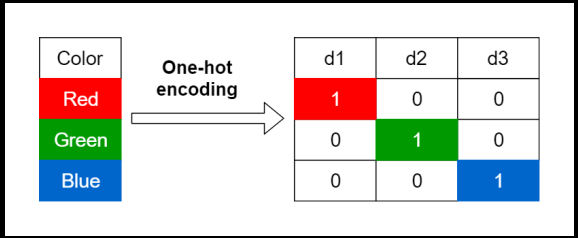


- 1. It uses an array of binary columns to represent each entity by a unique bit set to 1 while all others are 0.
- 2. Treats each entity as independent and orthogonal.
- 3. Encoded vectors are easily interpretable.
- 4. Does not require any learning.
- 5. Increases the data dimensionality by creating a new binary column for each category.
- 6. Inefficient and sparse when dealing with large number of categorical features.

[a]

[a] Source: [View link]

One-hot Encoding

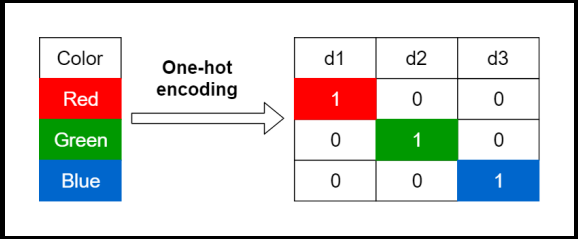


- 1. It uses an array of binary columns to represent each entity by a unique bit set to 1 while all others are 0.
- 2. Treats each entity as independent and orthogonal.
- 3. Encoded vectors are easily interpretable.
- 4. Does not require any learning.
- 5. Increases the data dimensionality by creating a new binary column for each category.
- 6. Inefficient and sparse when dealing with large number of categorical features.

a

[a] Source: [View link]

One-hot Encoding

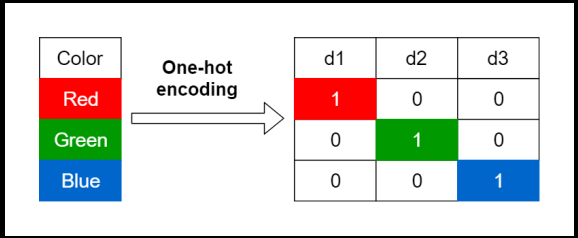


- 1. It uses an array of binary columns to represent each entity by a unique bit set to 1 while all others are 0.
- 2. Treats each entity as independent and orthogonal.
- 3. Encoded vectors are easily interpretable.
- 4. Does not require any learning.
- 5. Increases the data dimensionality by creating a new binary column for each category.
- 6. Inefficient and sparse when dealing with large number of categorical features.

a

[a] Source: [View link]

One-hot Encoding



- 1. It uses an array of binary columns to represent each entity by a unique bit set to 1 while all others are 0.
- 2. Treats each entity as independent and orthogonal.
- 3. Encoded vectors are easily interpretable.
- 4. Does not require any learning.
- 5. Increases the data dimensionality by creating a new binary column for each category.
- 6. Inefficient and sparse when dealing with large number of categorical features.

a

[a] Source: [View link]

Embedding

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

Embedding

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

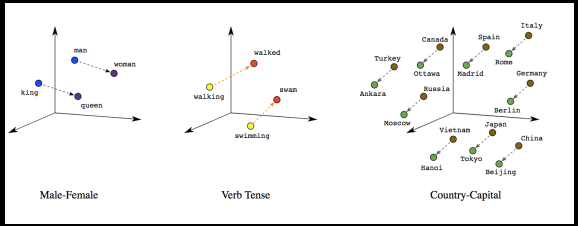
Embedding

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

Embedding

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

Word Embeddings

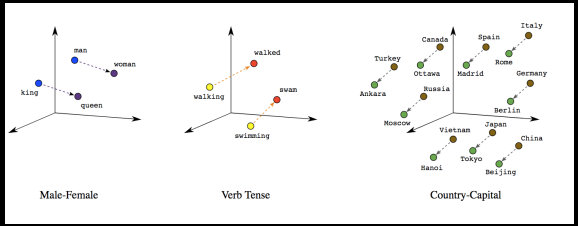


1. Word Embeddings encapsulates the word meaning in different contexts.

[a]

[a]Source: [View link]

Word Embeddings

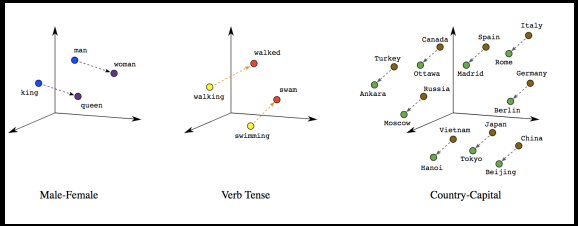


1. Word Embeddings encapsulates the word meaning in different contexts.
2. PCA on the Embeddings demonstrates how similar entities are clustered together in the embedded space.

[a]

[a]Source: [View link]

Word Embeddings

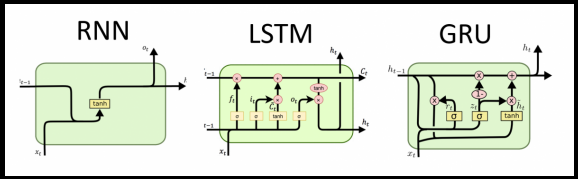


1. Word Embeddings encapsulates the word meaning in different contexts.
2. PCA on the Embeddings demonstrates how similar entities are clustered together in the embedded space.
3. Example: GloVe and Word2Vec

a

[a] Source: [View link]

Pre Tranformer Models - RNN, LSTM and GRU

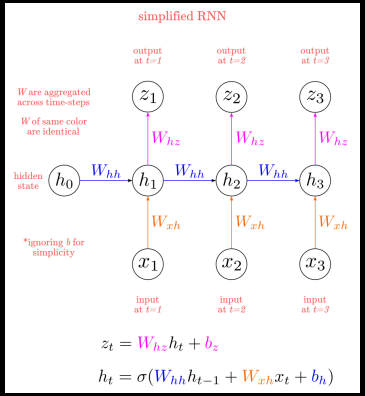


- Recurrent Neural Networks (RNN)
- Long Short-Term Memory (LSTM)
- Gated Recurrent Units (GRU)

[a]

[a] Source: [View link]

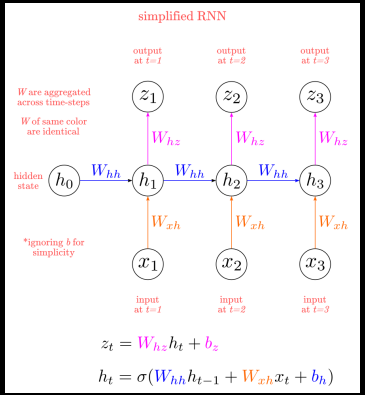
Recurrent Neural Network



[a]

[a]Source: [View link]

Recurrent Neural Network

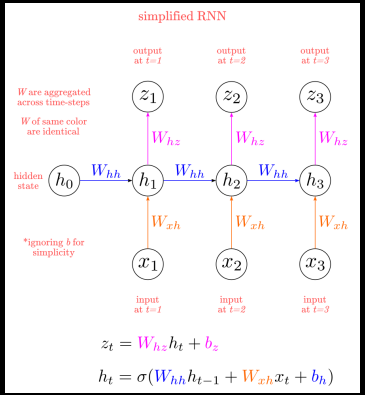


- If x_i is input, h_t is hidden state at time t and $W_{-,-}$ are connection weights,
- $h_z = \sigma(W_{hh}h_{t-1} + W_{xh}x_t + b_h)$
- $z_t = W_{hz}h_t + b_z$

[a]

[a]Source: [View link]

Recurrent Neural Network

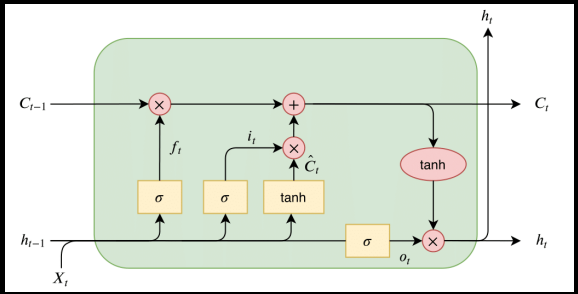


- If x_i is input, h_t is hidden state at time t and $W_{\_,\_}$ are connection weights,
- $h_z = \sigma(W_{hh}h_{t-1} + W_{xh}x_t + b_h)$
- $z_t = W_{hz}h_t + b_z$
- Issues: Vanishing and Exploding Gradients as the gradient of the error are backpropagated in time causing it to vanish if it is $\ll 1.0$ or explode if $\gg 1.0$

[a]

[a] Source: [View link]

Long Short-Term Memory Networks (LSTM)

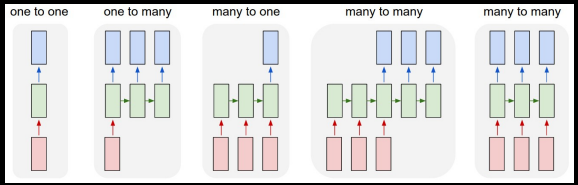


- LSTM use gating mechanisms (input, forget,output) to control the flow of information through the network over a longer period through the cell state C_t to prevent vanishing gradient problem.
- C_t transfers relevant information across different time steps.

[a]

[a] Source: [View link]

One2One to Many2Many Architectures

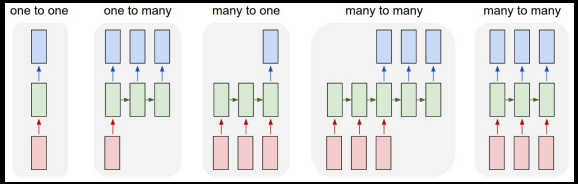


- One to One: Simple, vanilla model
- One to many: image-to-text conversion
- Many to One: Text to Image Generation
- Many to Many: Text translation

[a]

[a] Source: [View link]

Encoder-Decoder Architecture

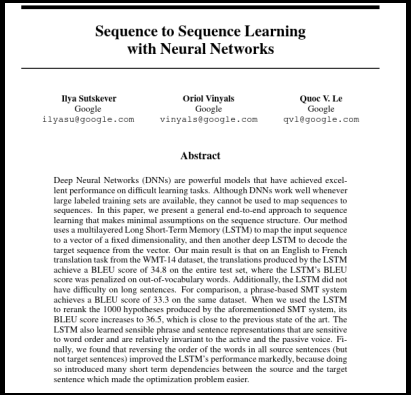


- What if the number of inputs differs from the number of outputs? For example, translation from one language to another?

[a]

[a] Source: [View link]

Sequence to Sequence Paper (2014)



Sequence to Sequence Learning with Neural Networks

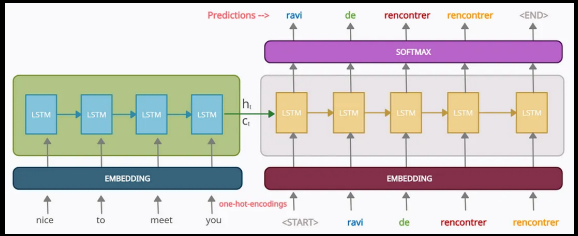
Ilya Sutskever Google ilyasu@google.com
Oriol Vinyals Google vinyals@google.com
Quoc V. Le Google qvl@google.com

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.

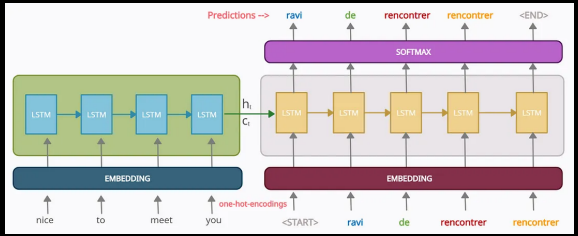
Sequence to Sequence Learning



1. LSTM generates fixed length context vector.

[a] Source: [View link]

Sequence to Sequence Learning

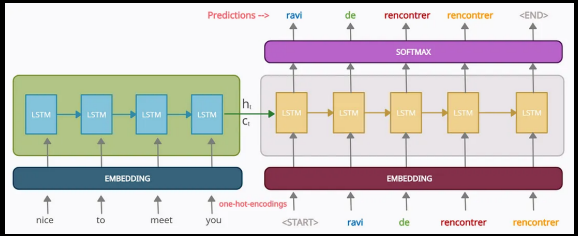


1. LSTM generates fixed length context vector.
2. The Decoder predicts a set of tokens that goes to a softmax function to predict the most probable token.

[a]

[a]Source: [View link]

Sequence to Sequence Learning

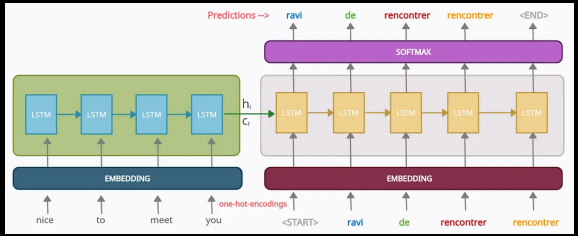


1. LSTM generates fixed length context vector.
2. The Decoder predicts a set of tokens that goes to a softmax function to predict the most probable token.
3. The most probable token found is now applied to the context vector to find the next token.

[a]

[a]Source: [View link]

Sequence to Sequence Learning



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]

Sequence to Sequence - Captures Context

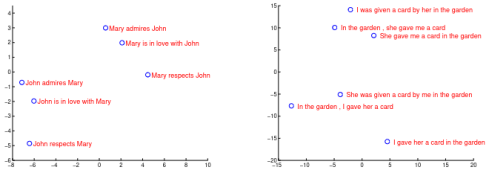
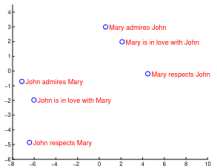


Figure 2: The figure shows a 2-dimensional PCA projection of the LSTM hidden states that are obtained after processing the phrases in the figures. The phrases are clustered by meaning, which in these examples is primarily a function of word order, which would be difficult to capture with a bag-of-words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

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

Sequence to Sequence - Captures Context



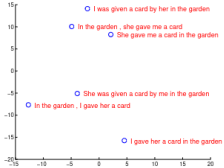
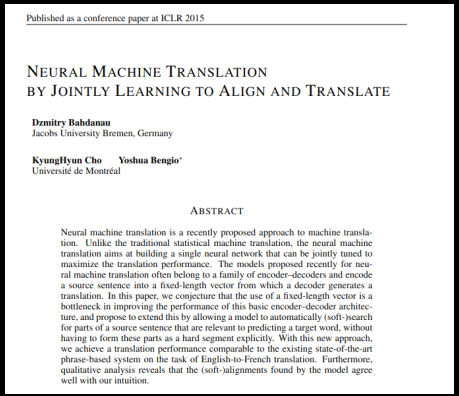


Figure 2: The figure shows a 2-dimensional PCA projection of the LSTM hidden states that are obtained after processing the phrases in the figures. The phrases are clustered by meaning, which in these examples is primarily a function of word order, which would be difficult to capture with a bag-of-words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

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

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

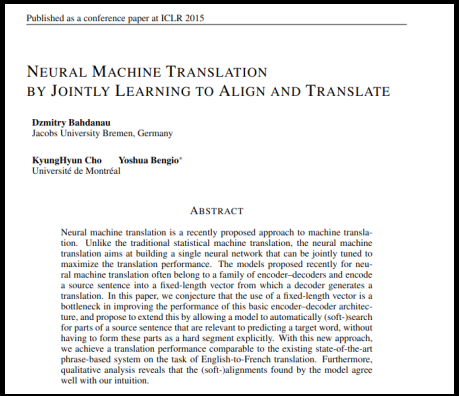
KyungHyun Cho **Yoshua Bengio***
Université de Montréal

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

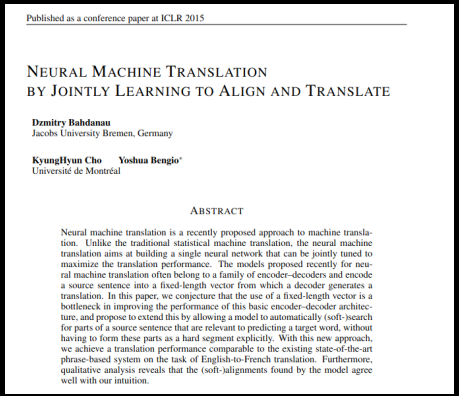
KyungHyun Cho **Yoshua Bengio***
Université de Montréal

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 sequence 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 sequence and its context into a single vector.
3. Instead of explicitly depending on a single Context Vector, the model searches for relevant parts of the sequence to get additional information for accurately predicting the target word.

Attention Mechanism, the 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:

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

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:

$$c_i = \sum_{j=1}^{T_x} \alpha_{ij} h_j. \quad (5)$$

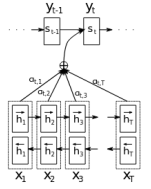
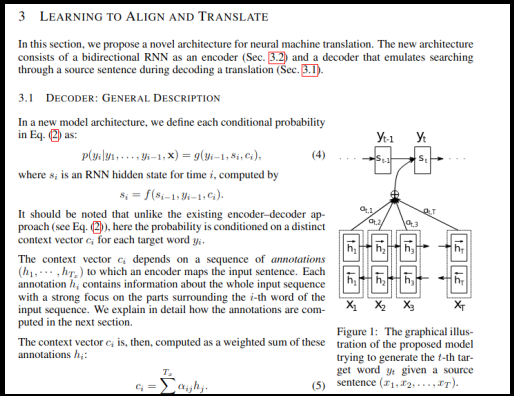


Figure 1: The graphical illustration of the proposed model trying to generate the t-th target word y_t given a source sentence $(x_1, x_2, \dots, x_T)$.

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

Attention Mechanism, the 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:

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

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.

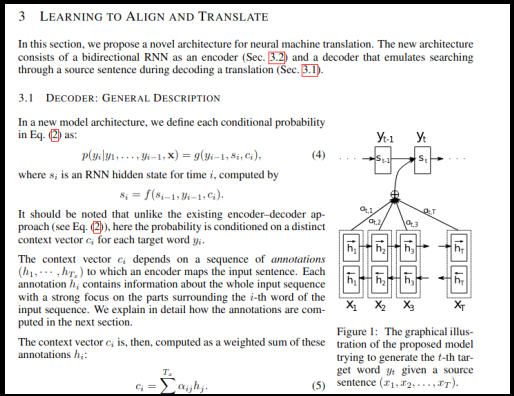
The context vector c_i is, then, computed as a weighted sum of these annotations h_i:

$$c_i = \sum_{j=1}^{T_x} \alpha_{ij} h_j. \quad (5)$$

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

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, the 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:

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

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.

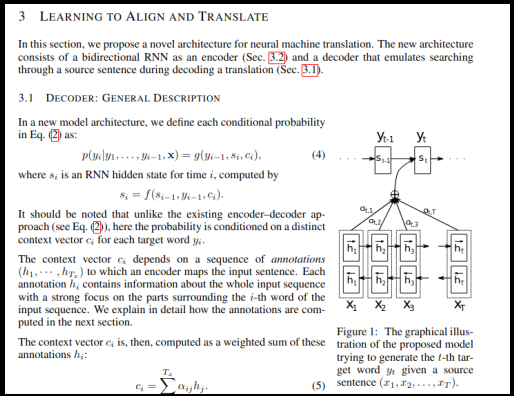
The context vector c_i is, then, computed as a weighted sum of these annotations h_i:

$$c_i = \sum_{j=1}^{T_x} \alpha_{ij} h_j. \quad (5)$$

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

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, the 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:

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

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.

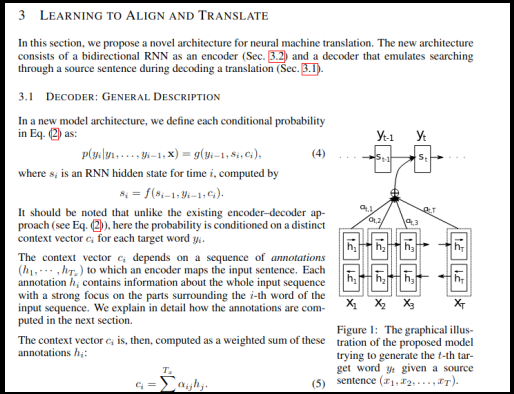
The context vector c_i is, then, computed as a weighted sum of these annotations h_i:

$$c_i = \sum_{j=1}^{T_x} \alpha_{ij} h_j. \quad (5)$$

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

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 simultaneously 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:

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

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.

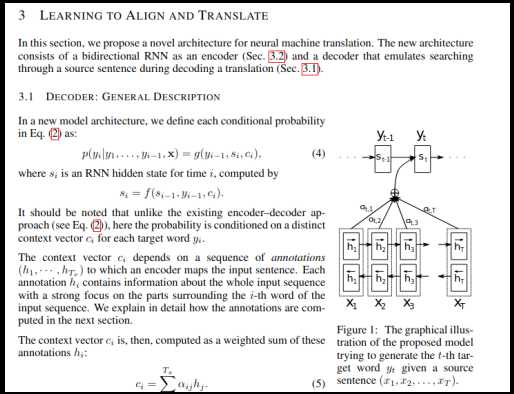
The context vector c_i is, then, computed as a weighted sum of these annotations h_i:

$$c_i = \sum_{j=1}^{T_x} \alpha_{ij} h_j. \quad (5)$$

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

- The 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:

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

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:

$$c_i = \sum_{j=1}^{T_x} \alpha_{ij} h_j. \quad (5)$$

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

- The 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 $X_1, X_2 ... X_t$, which means sequentially (one after another). This means the model is not scalable to be trained on large amounts of data.

Attention Recap

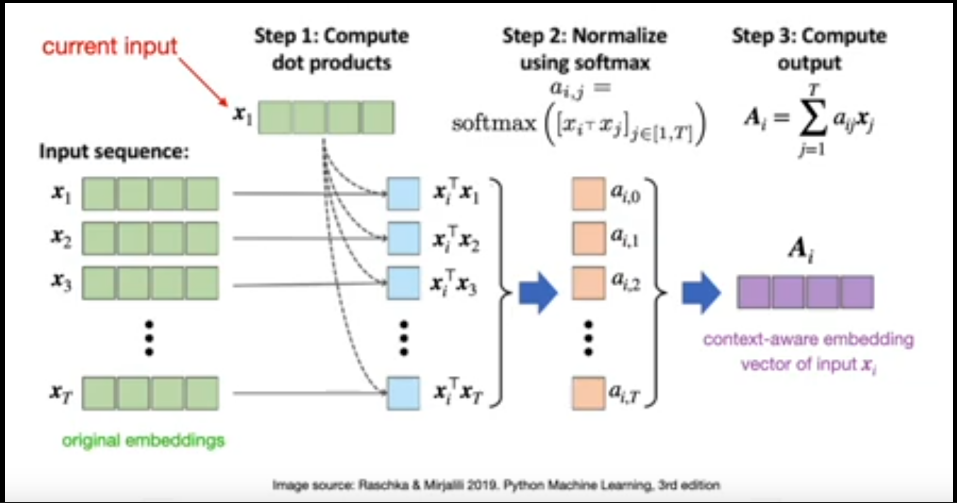


Image source: Raschka & Mirjalili 2019. Python Machine Learning, 3rd edition

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

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

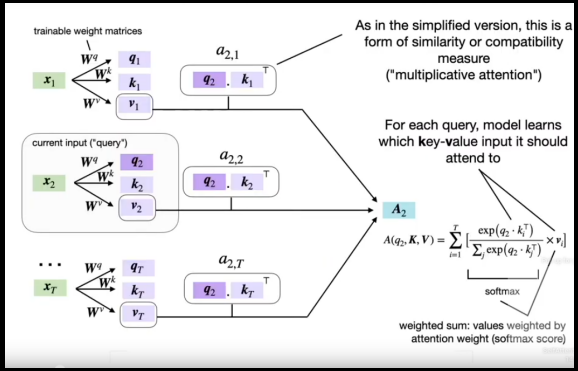
Aidan N. Gomez[*] [†]
University of Toronto
aidan@cs.toronto.edu

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*)

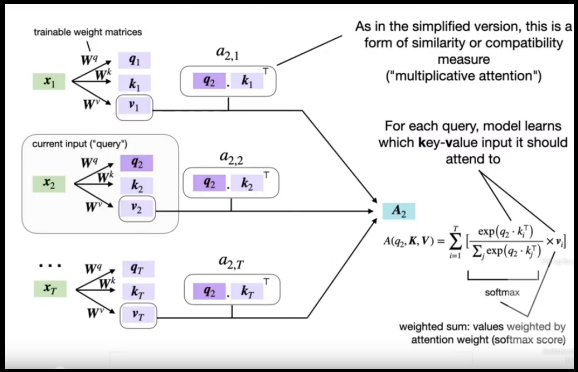
Self Attention



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

[a]Video Source

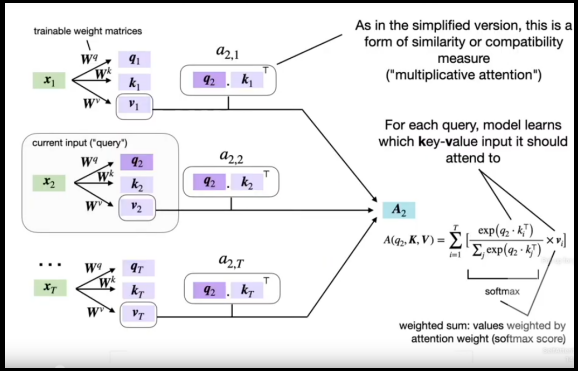
Self Attention



- -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]

[a]Video Source

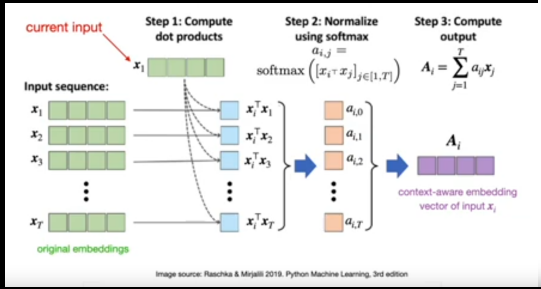
Self Attention

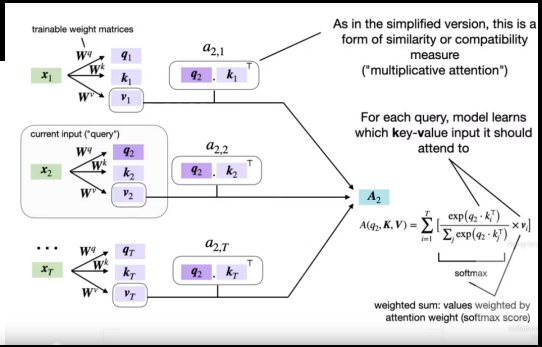


- -$Query, q = W_q.x_i$
- -$Key, k = W_k.x_i$
- -$Value, v = W_v.x_i$[a]
- - x is word embedding of dimension [1xN]
- - W is Weight Matrix of dimension [NxM]
- - q,k,v are thus having dimension [1xM]
- - $A(q_j, K, V) = \sum_{i=1}^{T} \frac{e^{q_j.k_i^T}}{\sum_j e^{q_j.k_j^T}} v_i$
- - dot product gives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

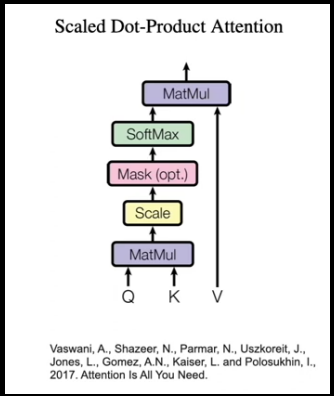
[a] Video Source

Attention vs Self Attention



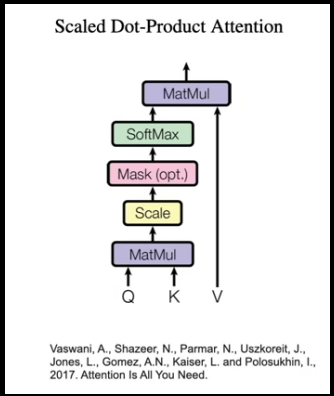


Self Attention



- - 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. $(Q, K, V) \rightarrow \mathbb{R}^{L \times M}$

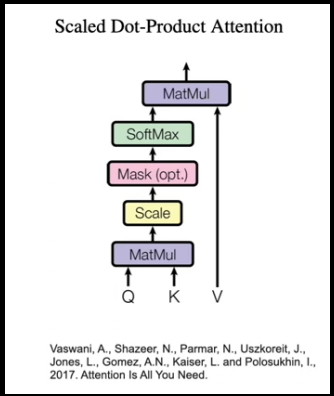
Self Attention



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

- - 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. $(Q, K, V) \rightarrow \mathbb{R}^{L \times M}$
- - Resulting A will be a matrix of dimension LxL and to prevent it from growing too large than the gradients, we scale it down by $\sqrt{M}$.

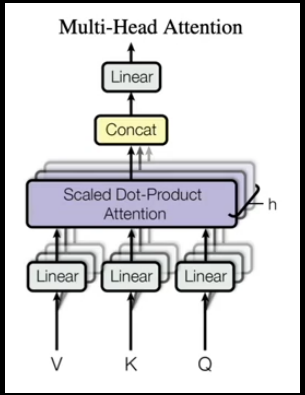
Self Attention



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

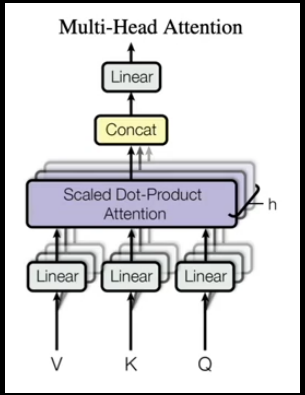
- - Since the computation can be done in parallel, the Attention for the whole A(Q, K, V) can be computed by treating Q as the vector of dimension L x N where L is the sequence Length. $(Q, K, V) \Rightarrow \mathbb{R}^{L \times M}$
- - Resulting A will be a matrix of dimension LxL and to prevent it 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!

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.

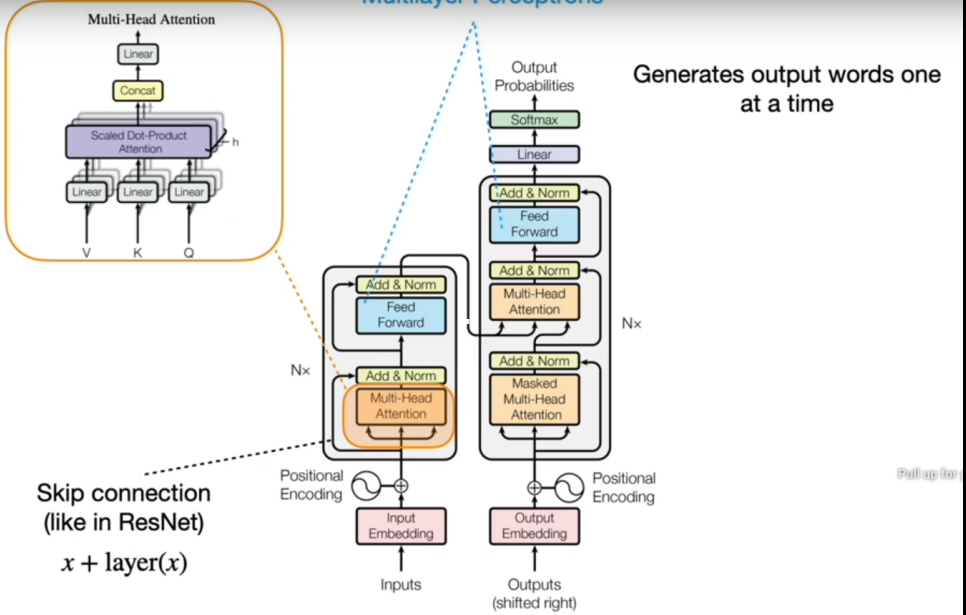


Figure 1: The Transformer - model architecture.

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

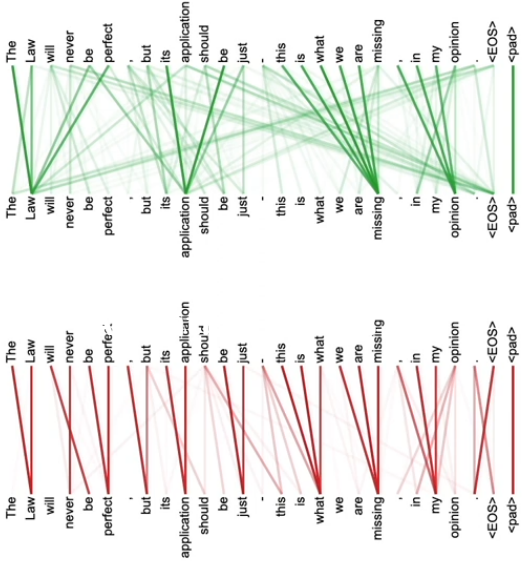
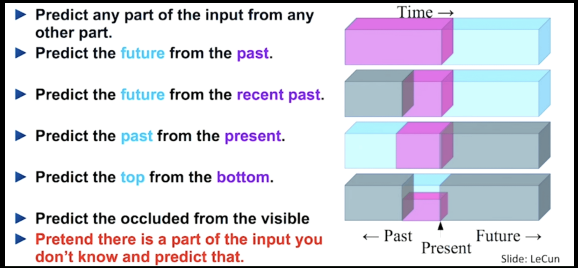


Figure 5: Many of the attention heads exhibit behaviour that seems related to the structure of the sentence. We give two such examples above, from two different heads from the encoder self-attention at layer 5 of 6. The heads clearly learned to perform different tasks.

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

Large Language Models

Self-Supervised Learning (SSL)

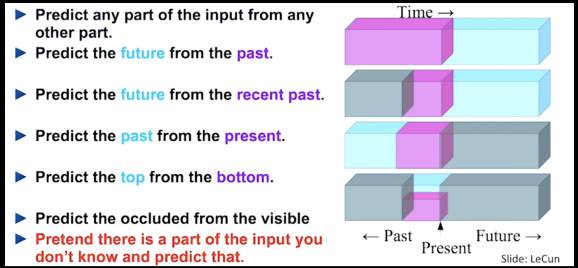


- Taking a corpus and creating labels by masking or structure analysis of the sequence. (QA)

[a]

[a]Source: [View link]

Self-Supervised Learning (SSL)

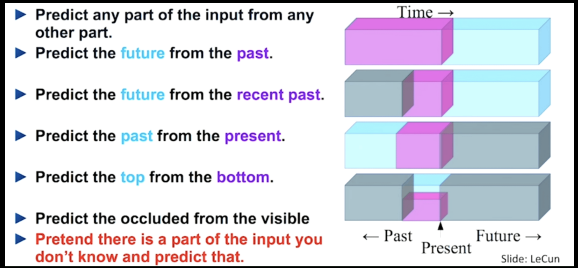


- Taking a corpus and creating labels by masking or structure analysis of the sequence. (QA)
- It empowers the model to learn underlying semantic features without introducing label bias.

[a]

[a]Source: [View link]

Self-Supervised Learning (SSL)

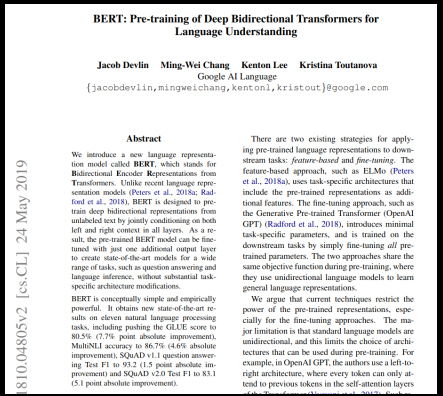


- Taking a corpus and creating labels by masking or structure analysis of the sequence. (QA)
- It empowers the model to learn underlying semantic features without introducing label bias.
- Generic AI: The self-supervised learning framework is one step closer to embedding human cognition in machines.

[a]

[a] Source: [View link]

Self Supervised Learning - Next Sentence Prediction (NSP)



BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding

Jacob Devlin **Ming-Wei Chang** **Kenton Lee** **Kristina Toutanova**
Google AI Language
{jacobdevlin,mingweichang,kentonl,kristout}@google.com

Abstract

We introduce a new language representation model called **BERT**, which stands for **B**idirectional **E**ncoder **R**epresentations from **T**ransformers. Unlike recent language representation models (Peters et al., 2018a; Radford et al., 2018), BERT is designed to pre-train deep bidirectional representations from unlabeled text by jointly conditioning on both left and right context in all layers. As a result, the pre-trained BERT model can be fine-tuned with just one additional output layer to create state-of-the-art models for a wide range of tasks, such as question answering and language inference, without substantial task-specific architecture modifications.

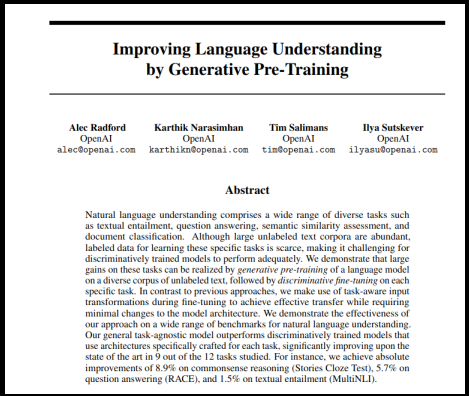
BERT is conceptually simple and empirically powerful. It obtains new state-of-the-art results on eleven natural language processing tasks, including pushing the GLUE score to 80.5% (7.7% point absolute improvement), MultiNLI accuracy to 86.7% (4.6% absolute improvement), SQuAD v1.1 question answering Test F1 to 93.2 (1.5 point absolute improvement) and SQuAD v2.0 Test F1 to 83.1 (5.1 point absolute improvement).

There are two existing strategies for applying pre-trained language representations to downstream tasks: *feature-based* and *fine-tuning*. The feature-based approach, such as ELMo (Peters et al., 2018a), uses task-specific architectures that include the pre-trained representations as additional features. The fine-tuning approach, such as the Generative Pre-trained Transformer (OpenAI GPT) (Radford et al., 2018), introduces minimal task-specific parameters, and is trained on the downstream tasks by simply fine-tuning *all* pre-trained parameters. The two approaches share the same objective function during pre-training, where they use unidirectional language models to learn general language representations.

We argue that current techniques restrict the power of the pre-trained representations, especially for the fine-tuning approaches. The major limitation is that standard language models are unidirectional, and this limits the choice of architectures that can be used during pre-training. For example, in OpenAI GPT, the authors use a left-to-right architecture, where every token can only attend to previous tokens in the self-attention layers

- Pick two simultaneous sentences from a document and a random sentence from the same or a different document, say sentence A, sentence B, and sentence C.
- Ask the model the relative position of sentence A with respect to sentence B?' – and the model outputs either IsNextSentence or IsNotNextSentence.

GPT (Generative Pre-trained Transformer)



Improving Language Understanding by Generative Pre-Training

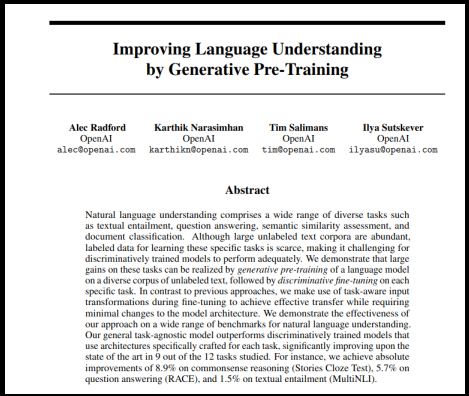
Alec Radford OpenAI alec@openai.com
Karthik Narasimhan OpenAI karthikn@openai.com
Tim Salimans OpenAI tim@openai.com
Ilya Sutskever OpenAI ilyasu@openai.com

Abstract

Natural language understanding comprises a wide range of diverse tasks such as textual entailment, question answering, semantic similarity assessment, and document classification. Although large unlabeled text corpora are abundant, labeled data for learning these specific tasks is scarce, making it challenging for discriminatively trained models to perform adequately. We demonstrate that large gains on these tasks can be realized by *generative pre-training* of a language model on a diverse corpus of unlabeled text, followed by *discriminative fine-tuning* on each specific task. In contrast to previous approaches, we make use of task-aware input transformations during fine-tuning to achieve effective transfer while requiring minimal changes to the model architecture. We demonstrate the effectiveness of our approach on a wide range of benchmarks for natural language understanding. Our general task-agnostic model outperforms discriminatively trained models that use architectures specifically crafted for each task, significantly improving upon the state of the art in 9 out of the 12 tasks studied. For instance, we achieve absolute improvements of 8.9% on commonsense reasoning (Stories Cloze Test), 5.7% on question answering (RACE), and 1.5% on textual entailment (MultiNLI).

- LLMs like GPT use Auto-regressive Language Modelling
- Predict the next word, having read all the previous ones.

GPT (Generative Pre-trained Transformer)



Improving Language Understanding by Generative Pre-Training

Alec Radford OpenAI alec@openai.com
Karthik Narasimhan OpenAI karthikn@openai.com
Tim Salimans OpenAI tim@openai.com
Ilya Sutskever OpenAI ilyasu@openai.com

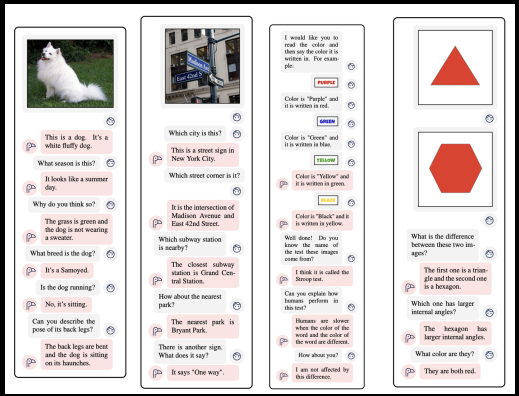
Abstract

Natural language understanding comprises a wide range of diverse tasks such as textual entailment, question answering, semantic similarity assessment, and document classification. Although large unlabeled text corpora are abundant, labeled data for learning these specific tasks is scarce, making it challenging for discriminatively trained models to perform adequately. We demonstrate that large gains on these tasks can be realized by *generative pre-training* of a language model on a diverse corpus of unlabeled text, followed by *discriminative fine-tuning* on each specific task. In contrast to previous approaches, we make use of task-aware input transformations during fine-tuning to achieve effective transfer while requiring minimal changes to the model architecture. We demonstrate the effectiveness of our approach on a wide range of benchmarks for natural language understanding. Our general task-agnostic model outperforms discriminatively trained models that use architectures specifically crafted for each task, significantly improving upon the state of the art in 9 out of the 12 tasks studied. For instance, we achieve absolute improvements of 8.9% on commonsense reasoning (Stories Cloze Test), 5.7% on question answering (RACE), and 1.5% on textual entailment (MultiNLI).

- LLMs like GPT use Auto-regressive Language Modelling
- Predict the next word, having read all the previous ones.
- A mask is used on top of the full sentence so that the attention heads can only see what was before in the text and not what is after. [a]

[a] The Paper

Large Multimodal Models (LMMs)

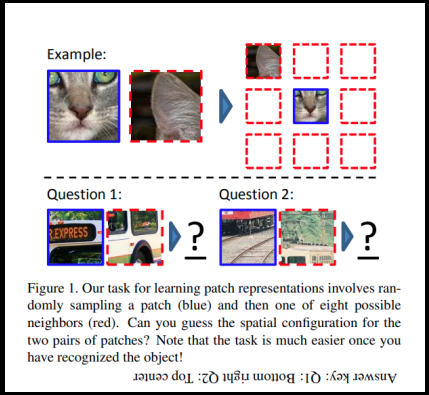


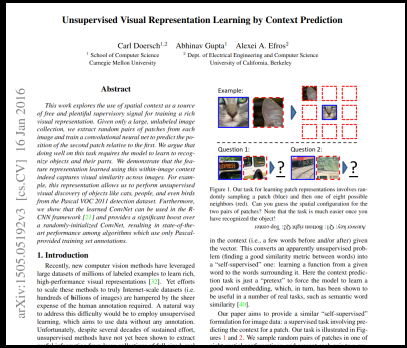
- Process multiple data types, including text, images, audio, and video
- Generate content across different modalities
- Integrate and interpret information from diverse sources simultaneously

[a]

[a] Source: [View link]

Self Supervised Learning - Patch Localisation





https://arxiv.org/pdf/1505.05192

a

[a]Source: [View link]

Contrastive Language-Image Pre-training (CLIP)

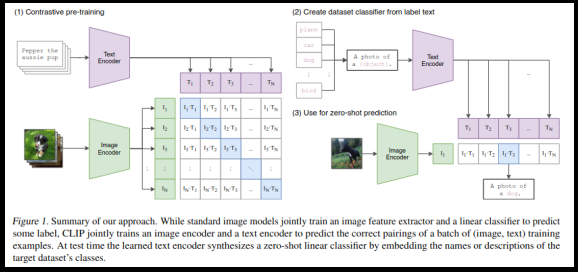
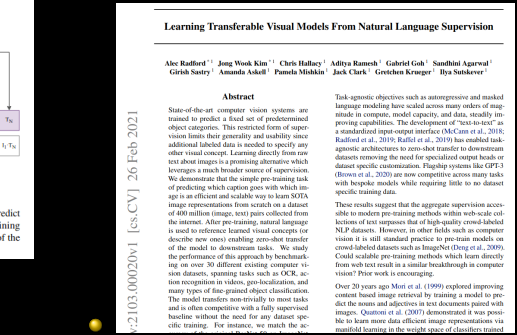


Figure 1. Summary of our approach. While standard image models jointly train an image feature extractor and a linear classifier to predict some label, CLIP jointly trains an image encoder and a text encoder to predict the correct pairings of a batch of (image, text) training examples. At test time the learned text encoder synthesizes a zero-shot linear classifier by embedding the names or descriptions of the target dataset's classes.



arXiv:2103.00020v1 [cs.CV] 26 Feb 2021

Learning Transferable Visual Models From Natural Language Supervision

Alec Radford [1] Jong Wook Kim [*1] Chris Hallacy [1] Aditya Ramesh [1] Gabriel Goh [1] Sandhini Agarwal [1] Girish Sastry [1] Amanda Askell [1] Pamela Mishkin [1] Jack Clark [1] Gretchen Krueger [1] Ilya Sutskever [1]

Abstract

State-of-the-art computer vision systems are trained to predict a fixed set of predetermined object categories. This restricted form of supervision limits their generality and usability since additional labeled data is needed to specify any other visual concept. Learning directly from raw text about images is a promising alternative which leverages a much broader source of supervision. We demonstrate that the simple pre-training task of predicting which caption goes with which image is an efficient and scalable way to learn SOTA image representations from scratch on a dataset of 400 million (image, text) pairs collected from the internet. After pre-training, natural language is used to reference learned visual concepts (or describe new ones) enabling zero-shot transfer of the model to downstream tasks. We study the performance of this approach by benchmarking on over 30 different existing computer vision datasets, spanning tasks such as OCR, action recognition in videos, geo-localization, and many types of fine-grained object classification. The model transfers non-trivially to most tasks and is often competitive with a fully supervised baseline without the need for any dataset specific training. For instance, we match the ac-

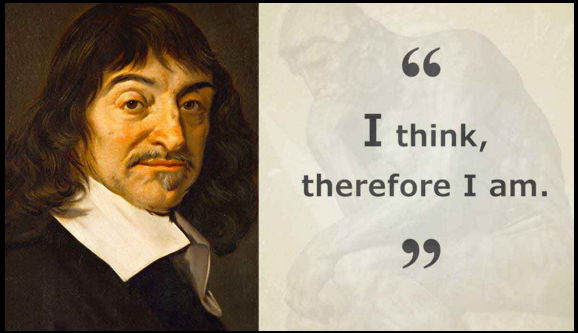
Task-agnostic objectives such as autoregressive and masked language modeling have scaled across many orders of magnitude in compute, model capacity, and data, steadily improving capabilities. The development of "text-to-text" as a standardized input-output interface (McCann et al., 2018; Radford et al., 2019; Raffel et al., 2019) has enabled task-agnostic architectures to zero-shot transfer to downstream datasets removing the need for specialized output heads or dataset specific customization. Flagship systems like GPT-3 (Brown et al., 2020) are now competitive across many tasks with bespoke models while requiring little to no dataset specific training data.

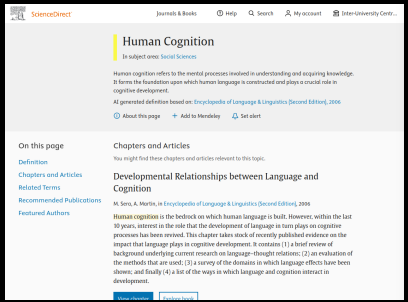
These results suggest that the aggregate supervision accessible to modern pre-training methods within web-scale collections of text surpasses that of high-quality crowd-labeled NLP datasets. However, in other fields such as computer vision it is still standard practice to pre-train models on crowd-labeled datasets such as ImageNet (Deng et al., 2009). Could scalable pre-training methods which learn directly from web text result in a similar breakthrough in computer vision? Prior work is encouraging.

Over 20 years ago Mori et al. (1999) explored improving content based image retrieval by training a model to predict the nouns and adjectives in text documents paired with images. Quattoni et al. (2007) demonstrated it was possible to learn more data efficient image representations via manifold learning in the weight space of classifiers trained

[a] CLIP paper link

Human Cognition



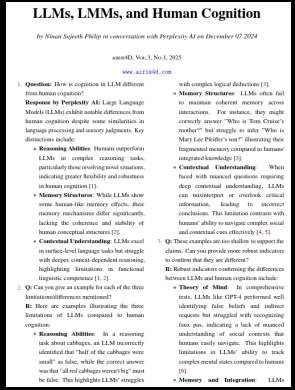


- Rene Descartes most famous dictums: ”Cogito, Ergo Sum”.

[a]

[a] Source: [View link]

LLMs and Human Cognition



LLMs, LMMs, and Human Cognition

by Ninan Sajeeth Philip in conversation with Perplexity AI on December 07 2024

airis4D, Vol.3, No.1, 2025
www.airis4d.com

1 **Question:** How is cognition in LLM different from human cognition?
Response by Perplexity AI: Large Language Models (LLMs) exhibit notable differences from human cognition despite some similarities in language processing and sensory judgments. Key distinctions include:
- **Reasoning Abilities**: Humans outperform LLMs in complex reasoning tasks, particularly those involving novel situations, indicating greater flexibility and robustness in human cognition [1].
- **Memory Structures**: While LLMs show some human-like memory effects, their memory mechanisms differ significantly, lacking the coherence and stability of human conceptual structures [2].
- **Contextual Understanding**: LLMs excel in surface-level language tasks but struggle with deeper, context-dependent reasoning, highlighting limitations in functional linguistic competence [1, 2].

2 **Q:** Can you give an example for each of the three limitations/differences mentioned?
R: Here are examples illustrating the three limitations of LLMs compared to human cognition:
- **Reasoning Abilities**: In a reasoning task about cabbages, an LLM incorrectly identified that "half of the cabbages were small" as false, while the correct answer was that "all red cabbages weren't big" must be false. This highlights LLMs' struggles with complex logical deductions [3].
- **Memory Structures**: LLMs often fail to maintain coherent memory across interactions. For instance, they might correctly answer "Who is Tom Cruise's mother?" but struggle to infer "Who is Mary Lee Pfeiffer's son?" illustrating their fragmented memory compared to humans' integrated knowledge [3].
- **Contextual Understanding**: When faced with nuanced questions requiring deep contextual understanding, LLMs can misinterpret or overlook critical information, leading to incorrect conclusions. This limitation contrasts with humans' ability to navigate complex social and contextual cues effectively [4, 5].

3 **Q:** These examples are too shallow to support the claims. Can you provide more robust indicators to confirm that they are different?
R: Robust indicators confirming the differences between LLMs and human cognition include:
- **Theory of Mind**: In comprehensive tests, LLMs like GPT-4 performed well identifying false beliefs and indirect requests but struggled with recognizing faux pas, indicating a lack of nuanced understanding of social contexts that humans easily navigate. This highlights limitations in LLMs' ability to track complex mental states compared to humans [6].
- **Memory and Integration**: LLMs

airis4D Jornal Link

Questions

Questions

- Thank You