AIは意味をどのように扱っているのか？

意味の分散表現論の系譜

意味の分散表現論の系譜
Agenda

はじめに

Part 1 意味の分散表現論のはじまり

Part 2 大規模言語モデルへ

参考資料

- YouTube:　Maruyama Lectures
  https://www.youtube.com/@MaruyamaLectures

- MaruLaboサイト
  https://www.marulabo.net/


はじめに

ChatGPTの訴求力はどこから来るのか？

ChatGPTが、多くの人に強い印象を与えるのには理由があります。

それは、ChatGPTが、機械が自在にことばを操る能力を獲得したように見えるからです。これまでは、ことばを操る能力はもっぱら人間だけのものだと考えられていましたから。

そして、機械が人間並の言語能力を獲得し始めたという認識は、正しいものだと僕は考えています。

人工知能の「言語能力」の画期的進歩

今日の「人工知能」は、その「言語能力」で、画期的な進歩を遂げています。

- ある言語の文を他の言語に「翻訳」すること。
- 言語を問わず膨大な文字データから「関連する」データを見つけ出すこと。
- 長い文章を短い文章に「要約」すること。
- ある話題に対して関連する話題を提供して「対話」を続けること。

「大規模言語モデル」の成長とその秘密

こうした能力は、「大規模言語モデル」と呼ばれる人工知能技術の登場とその成長によって初めて可能になりました。

その成長の秘密の鍵は、この技術が「ことばの意味」を、コンピュータ上で表現する方法を見つけたことにあります。

イメージの認識なら、イメージをピクセルの集まりとしてコンピュータ上で表現することは容易です。ただ、「ことばの意味」を、コンピュータの上で表現せよといわれたら、みなさんはどんな方法をイメージしますか？

「意味の近さ」を表現できる

Googleの検索は、膨大なネット上の文字データから、基本的には特定の「文字列」を探すものです。「意味」を「検索」しているわけではありません。

「大規模言語モデル」では、「意味」がコンピュータ上に表現されています。

この「意味の表現」では、「意味が同じ」ことだけでなく「意味が近い」ことが表現できます。「意味の近さ」が表現できるということは、実践的にはとても大事です。

「大規模言語モデル」が獲得した「意味の表現」の力

「翻訳」でしたら「意味が同じ」ことが重要です。話題の「関連性」を見つけるとか「要約」のケースでは「意味が近い」ことがポイントになります。

こうした判断をコンピュータが自由に行えるようになったことが、コンピュータの「言語能力」の飛躍をもたらしました。それらは、「大規模言語モデル」が獲得した「意味の表現」の力によるものです。

「意味の分散表現」技術の現在

今回のセミナーは、この「意味の分散表現」技術にフォーカスして、大規模言語モデルにいたるこの技術の系譜を歴史的に振り返ったものです。

多くの技術は、理論的な基礎を持ちます。その理論は数学的形式を取ります。ただ、全ての技術が理論に導かれて発展してきたわけではありません。原爆を作れたのは、物理学の理論によるものでしたが、熱力学や統計力学が蒸気機関を生み出したわけではありません。

21世紀の「人工知能」技術は、18世紀の蒸気機関に似ています。この技術の背後にある理論 -- それはまだ多くの謎に満ちています – を研究するフェーズが生まれているのです。

意味の分散表現論の系譜
Agenda

Part 1 意味の分散表現論のはじまり

- 2003年：Bengioの「次元の呪い」と語の特徴の分散表現
- 2006年：HintonのAuto Encoder -- 意味的ハッシング
- 2011年：RNNによる文の生成
- 2013年：Word2Vec -- 語の意味ベクトル
- 2014年：Sequence to Sequence -- 文の意味ベクトル
- 2015年：RNNの不思議な力 -- RNNは文法を理解している

意味の分散表現論の系譜
Agenda

Part 2 大規模言語モデルへ

- 2016年：Attention Mechanism
- 2016年：Google ニューラル機械翻訳
- 2017年：Transformer
- 2019年：BERT


Part 1 意味の分散表現論のはじまり

Bengioの「次元の呪い」と語の特徴の分散表現

2003年

Scaling to Very Very Large Corpora for Natural Language Disambiguation

Michele Banko and Eric Brill
Microsoft Research
1 Microsoft Way
Redmond, WA 98052 USA
{mbanko,brill}@microsoft.com

Abstract

The amount of readily available on-line text has reached hundreds of billions of words and continues to grow. Yet for most core natural language tasks, algorithms continue to be optimized, tested and compared after training on corpora consisting of only one million words or less. In this paper, we evaluate the performance of different

potentially large cost of annotating data for those learning methods that rely on labeled text.

The empirical NLP community has put substantial effort into evaluating performance of a large number of machine learning methods over fixed, and relatively small, data sets. Yet since we now have access to significantly more data, one has to wonder what conclusions that have been drawn on small data sets may carry over when these learning methods are trained using much larger corpora.

In this paper, we present a study of the

We propose that a logical next step for the research community would be to direct efforts towards increasing the size of annotated training collections

初期の言語への統計的アプローチの失敗

初期の、大量の言語データを統計的に処理すれば、言語の特質がわかるだろうという楽観的な見通しは、うまくいかなかった。

例えば、次の文の下線部に、三つの単語{to, two, too}から、一つ選んで文章を作るという問題を考えよう。

For breakfast I ate _____eggs.

しかし、機械に10億語もの用例集 "Very Very Large Corpora"を学ばせても、正解率が、100%に届かないのだ。

https://aclanthology.org/P01-1005.pdf

A Neural Probabilistic Language Model

Yoshua Bengio et al.

http://goo.gl/977AQp

2003年

featureの分散表現で、「次元の呪い」と戦う

この論文で、彼は、次のような方法を提案する。

1. 語彙中のそれぞれの語に、R^mに実数値の値を持つ、分散した語の特徴ベクトル(word feature vector)を対応づける。
2. 語の並びの結合確率関数を、この並びの中の語の特徴ベクトルで表現する。
3. 語の特徴ベクトルとこの確率関数のパラメーターを、同時に学習する。

要は、語の「特徴ベクトル」という、語の「意味」の対応物を導入しようということだと僕は理解している。こうしたアプローチは、Tomas Mikolovらの **Word2Vec** に受け継がれていく。

少し乱暴にまとめてみる

- 「文」全体を相手にすると、その数はあまりに多すぎて、いくら統計的手法を使っても、言語の特徴をとらえるのは難しい。「次元の呪い」で勝ち目はない。それより遥かに数の少ない「語」の特徴づけから始めよう。「語」から「文」を攻めよう。
- それにしても「語」の特徴が一つの数字で表されるとは思えない。m個の数字の組 すなわちm次元のベクトルとして、「語」の特徴を表すことにしよう。
- 「文」は「語」の並びである。「文」が、「語」の並びとしてある確率的特徴を持つなら、それは「語」の特徴と結びつけることができるはずだ。
- コンピュータは、語の特徴ベクトルと「語」の並びの結合確率関数のパラメーターを、同時に学習すればいい。

HintonのAuto Encoder
意味的ハッシング

2006年

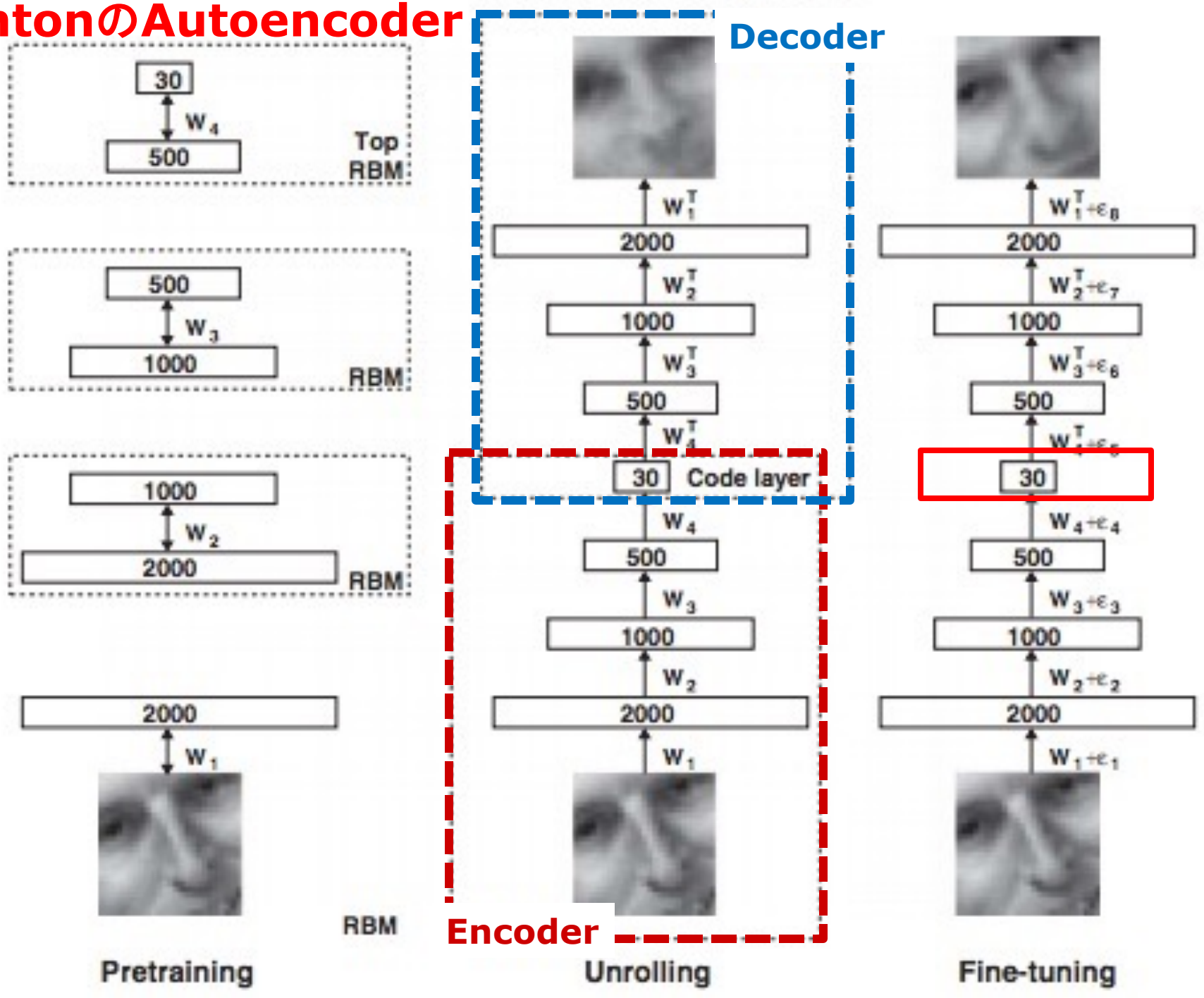
Reducing the Dimensionality of Data with Neural Networks

G. E. Hinton et al.

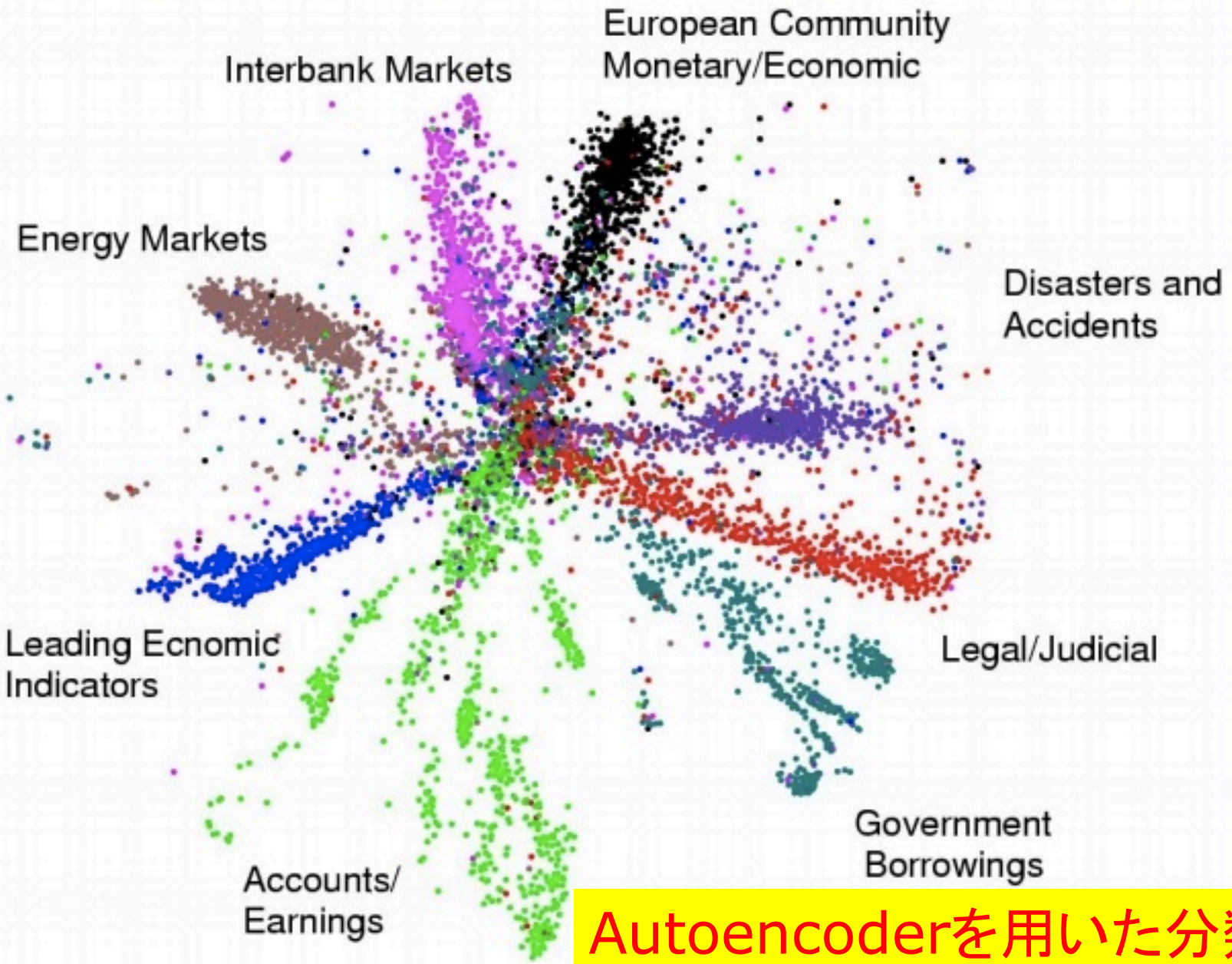
https://www.cs.cmu.edu/~tom/10701_11/slides/DeepNets_science2006.pdf

2006年

HintonのAutoencoder



First compress all documents to 2 numbers.
Then use different colors for different document categories



Autoencoderを用いた分類

Semantic hashing（意味的ハッシング）

- 重要なことは、「画像」と「書籍」では、対象のデータの性質はまるで異なるのだが、Autoencoderは、そのいずれに対しても、高次元のデータを低次元のデータに変換しているということである。別の言葉で言えば、**それは、対象の高次元のデータから、低次元のデータを、元の情報のエッセンスとして取り出しているのである。**
- Hintonは、こうしたAutoencoderの働きを、**Semantic hashing（意味的ハッシング）**と呼んでいる。
- SHA-1のようなハッシングでは、ハッシュ化されたデータから元のデータを復元することは不可能なのだが、Semantic hashingされたデータは、データの次元は低いものの、元の情報の中核部分を保持している。

RNNによる文の生成

2011年

Generating Text with Recurrent Neural Networks

Ilya Sutskever et al.

http://www.cs.utoronto.ca/~ilya/pubs/2011/LANG-RNN.pdf

2011年

RNNによる文章の生成

Googleの Ilya Sutskeverは、文字数が5億文字にものぼるテキストを長い時間をかけてRecurrent Neural Nets に学習させ、次のページのような文章を生成することができた。

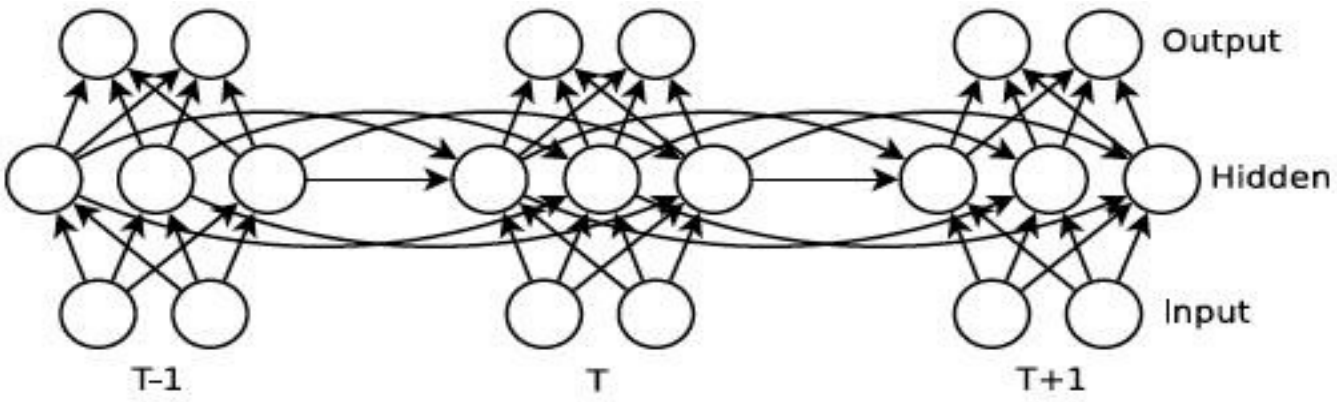


Figure 1. A Recurrent Neural Network is a very deep feedforward neural network whose weights are shared across time. The non-linear activation function used by the hidden units is the source of the RNN's rich dynamics.

WikiPediaで学習して生成された文章の例

The meaning of life is the tradition of the ancient human reproduction: it is less favorable to the good boy for when to remove her bigger. In the show’s agreement unanimously resurfaced. The wild pasteured with consistent street forests were incorporated by the 15th century BE. In 1996 the primary rapford undergoes an effort that the reserve conditioning, written into Jewish cities, sleepers to incorporate the .St Eurasia that activates the population.

http://goo.gl/vHRHSn

New York Timesで学習して生成された文章の例

while he was giving attention to the second advantage of school building a 2-for-2 stool killed by the Cultures saddled with a halfsuit defending the Bharatiya Fernall ’s office . Ms . Claire Parters will also have a history temple for him to raise jobs until naked Prodiena to paint baseball partners , provided people to ride both of Manhattan in 1978 , but what was largely directed to China in 1946 , focusing on the trademark period is the sailboat yesterday and comments on whom they obtain overheard within the 120th anniversary , where .....

http://goo.gl/vHRHSn

Word2Vec --「語の意味ベクトル」

2013年

MikolovらのWord2Vec論文が出るのは、Benjioの論文から10年後の2013年だった。

Linguistic Regularities in Continuous Space Word Representations

Tomas Mikolov et al.

https://www.microsoft.com/en-us/research/wp-content/uploads/2016/02/rvecs.pdf

2013年

Word2Vec論文

2013年に、Google（当時）のTomas Mikolovらは、語が埋め込まれたベクター空間が、言語学的に（文法的にも、意味論的にも）面白い性質を持っていることを発見する。

Tomas Mikolov,Wen-tau Yih,Geoffrey Zweiget

Linguistic Regularities in Continuous Space Word Representations

https://www.microsoft.com/en-us/research/wp-content/uploads/2016/02/rvecs.pdf

Word2Vec 二つ目の論文とコード公開

Jeff Deanが先の論文に興味を持ち、共同で研究を始め二つ目の論文を発表。

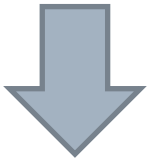
Tomas Mikolov,Kai Chen,Greg Corrado,Jeffrey Dean
Efficient Estimation of Word Representations in Vector Space
https://arxiv.org/pdf/1301.3781.pdf

Google Codeに、オープンソースとして公開され、大きな関心を集める。
https://code.google.com/p/word2vec/

どんな語が、与えられた語の近くに埋め込まれるか？

- 似た意味を持つ言葉は、似たベクトルを持つ。
- 似た言葉で置き換えても、正しい文は、正しい文に変わる。

“a **few** people sing well”　　正しい文



“a ***couple*** people sing well”　　正しい文

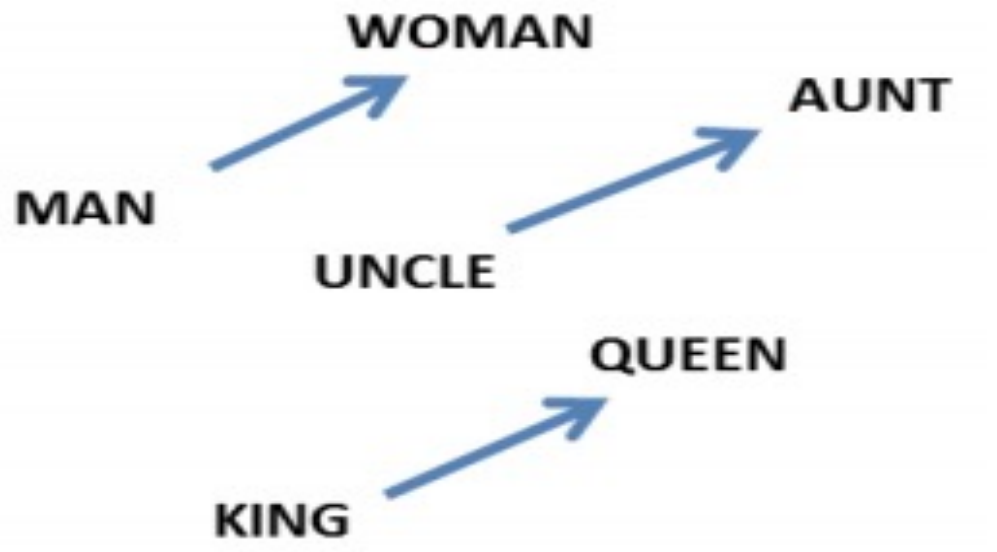
- 意味が似ていなくても、同じクラスの言葉で置き換えても、正しい文は、正しい文に変わる。

“the **wall** is **blue**” ⟶ “the ***ceiling*** is ***red***”

意味を変換するベクトルは共通？

Word Embeddingは、もっと面白い性質を持つ。下の図のように、男性から女性へのベクトルがあるように見える。

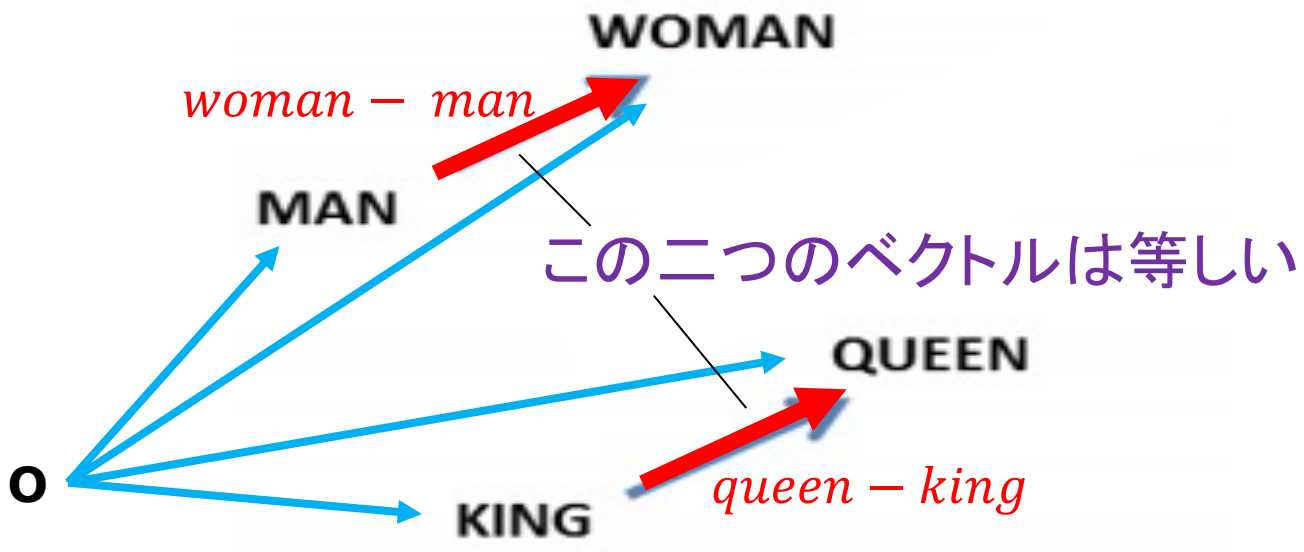
W(“woman”)−W(“man”) ≃ W(“aunt”)−W(“uncle”)
W("woman")−W("man") ≃ W("queen")−W("king")



意味を変換するベクトルは共通？
Vector Offset Method

Word Embeddingは、もっと面白い性質を持つ。下の図のように、男性から女性へのベクトルがあるように見える。

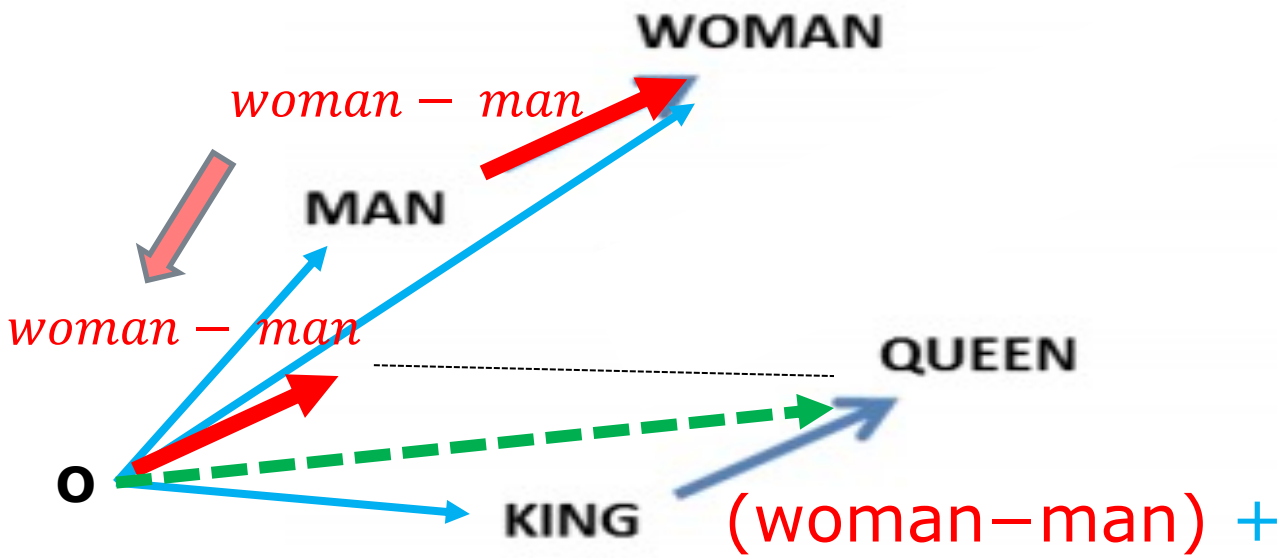
W(“woman")−W(“man") ≃ W(“queen")−W(“king")



King - Man + Woman = Queen
Vector Offset Method

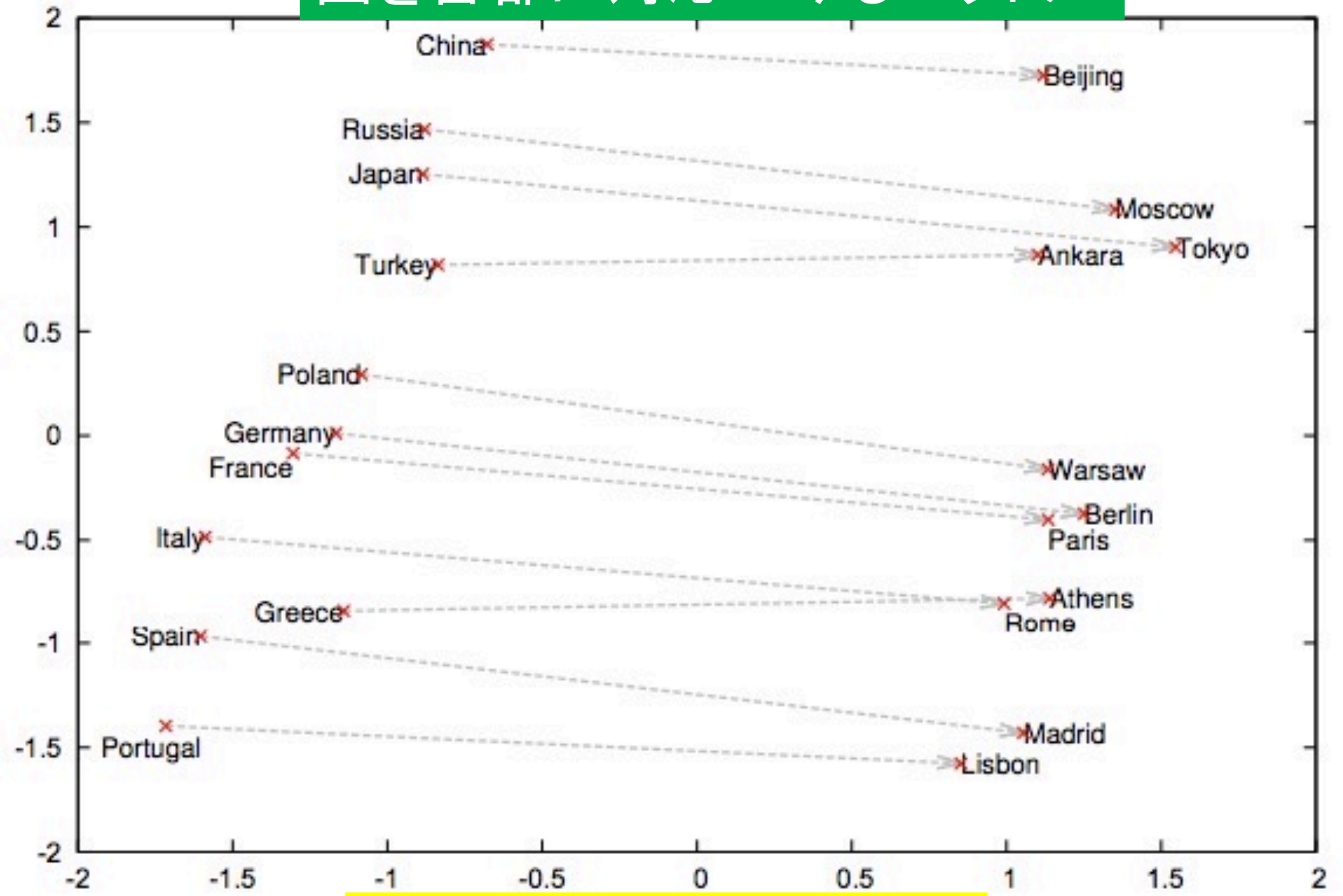
次のような変形もできる。

W(“woman")–W(“man") + W(“king") ≃ W(“queen")



(woman–man) + king ≃ queen

国を首都に対応づけるベクトル



http://arxiv.org/pdf/1310.4546.pdf

「語の意味へのベクトル：で重要なこと
語の意味の近さを定義できる

「語の意味のベクトル表現」でもっとも重要なことは、このアプローチによって、語の意味の近さを定義できることである。

語vと語wのベクトル表現を$\vec{v}$と$\vec{w}$する。この時、

語vと語wの意味の近さ$Similarity(v,w)$を、ベクトル$\vec{v}$と$\vec{w}$の内積$\vec{v}\cdot\vec{w}$で定義する。

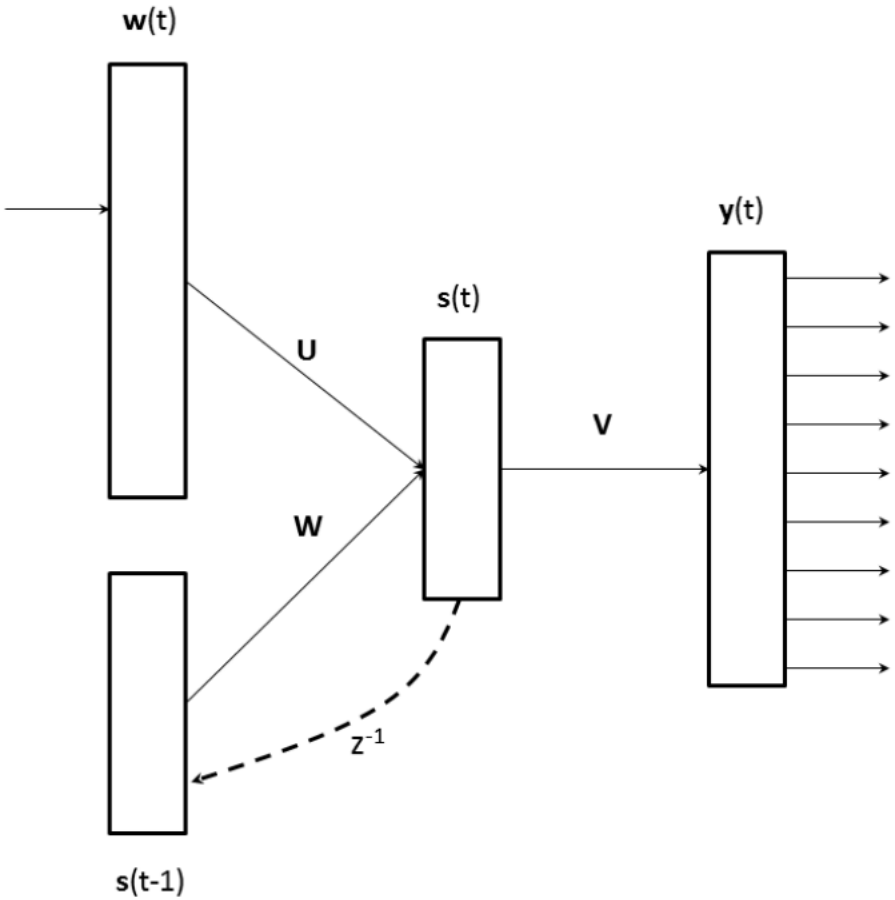
$$Similarity(v,w)=\vec{v}\cdot\vec{w}=|\overrightarrow{v}||\overrightarrow{w}|cos\theta$$

ここに、θは、ベクトル$\vec{v}$と$\vec{w}$のなす角である。これを、cosine-similarityと呼ぶ。

語はどのようにベクトルに変換されるのか

WordをVectorに変える方法

第一論文では、次のようなRNNが使われている。



$$\mathbf{s}(t) = f\left(\mathbf{U}\mathbf{w}(t) + \mathbf{W}\mathbf{s}(t-1)\right)$$

$$\mathbf{y}(t) = g\left(\mathbf{V}\mathbf{s}(t)\right),$$

$$f(z) = \frac{1}{1+e^{-z}}$$

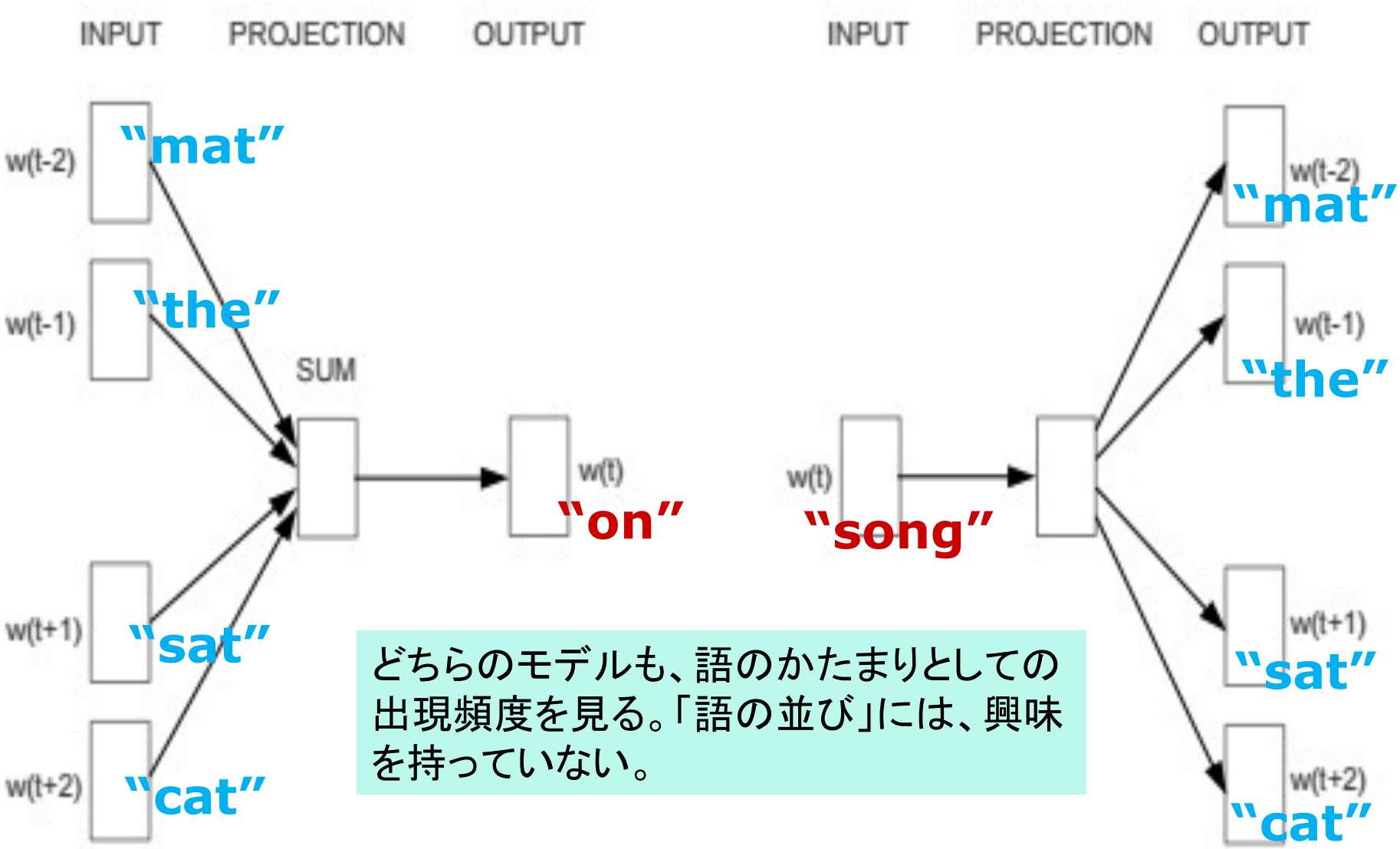
$$g(z_m) = \frac{e^{z_m}}{\sum_k e^{z_k}}$$

WordをVectorに変える方法

第二論文で使われている方法は、次の二つである。

- **CBOW(Continuus Bag-of-Word”) モデル**
  複数の語の集まりから、一緒に出現しそうな一つの語の確率を調べる。

- **Skip-gram モデル**
  一つの語が与えられた時、一緒に出現しそうな複数の語の確率を調べる。



Continuous Bag-of-Words

CBOW

Skip-gram

http://arxiv.org/pdf/1301.3781.pdf

語の意味の分散モデル を作る
Tai-Danaeの説明

例えば好きな本でもいい、ある固定したコーパスがあったとしよう。
そこから、context words $\{w_1, \ldots, w_n\}$ と呼ばれる語の集まりを選ぶ。それは、コーパス内のすべての言葉でもいいし、その一部でもいい。この$\{w_1, \ldots, w_n\}$ のw_iをベクトル空間Vの i 番目の標準的な基底と考えるのだ。

そうすれば、コーパス内のすべての語は、次のようなcontext wordの線形結合で表されるベクトル表現を持つことになる。

$$\boldsymbol{w} = \sum_{i=1}^{n} c_i \boldsymbol{w_i}$$

この係数c_iは、コーパス内で、語wが語w_iの近くに現れた回数を示す実数である。

語の意味の分散モデルの例

{sweet, green, furry} という語を含むある本があったとしよう。この三つの語をこのコーパスのcontext word に選んだら、これを基底として、次のようにベクトルで表す。

$$\mathbf{sweet} = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \qquad \mathbf{green} = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} \qquad \mathbf{furry} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

この時、この本の中の語 banana, puppy, fruit は、次のように表される。

$$\mathbf{banana} = \begin{bmatrix} 21 \\ 9 \\ 0 \end{bmatrix} \qquad \mathbf{puppy} = \begin{bmatrix} 8 \\ 1 \\ 32 \end{bmatrix} \qquad \mathbf{fruit} = \begin{bmatrix} 43 \\ 19 \\ 0 \end{bmatrix}$$

語の意味の分散モデルの例

別の言葉で言えば、我々は、コーパスからのデータをこれらの語を三次元のベクトル空間に埋め込むために利用したのである。

語 banana の意味は (21, 9, 0)、語 puppy の意味は (8, 1, 32)、語 fruitの意味は (43, 19, 0) ということになる。

Sequence to Sequence
文の意味ベクトル

2014年

Sequence to Sequence Learning with Neural Networks

Ilya Sutskever et al.

https://arxiv.org/pdf/1409.3215.pdf

2014年

文の意味ベクトルの発見

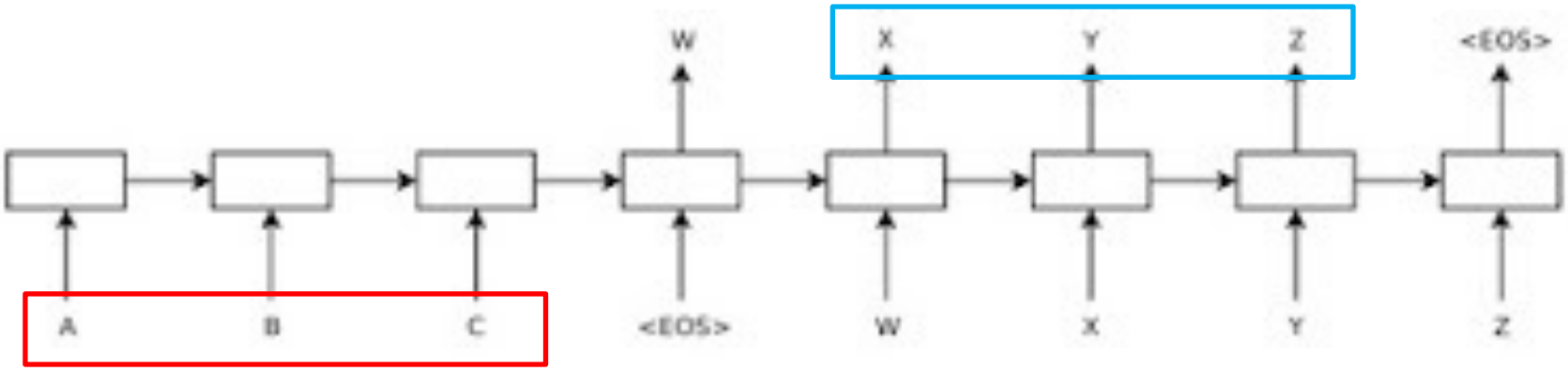
2014年に、Ilya Sutskever らは、シーケンスをシーケンスに変換するRNN(LSTM)の能力が、機械翻訳に応用できるという論文を発表します。

「我々の方法では、入力のシーケンスを固定次元のベクトルにマップするのに、多層のLong Short-Term Memory（LSTM）を利用する。その後、別の深いLSTMが、このベクトルから目的のシーケンスをデコードする。」

それでは、二つのSequence を結びつけているのはなんでしょう。それは二つのSequenceが「同じ意味」を持つということです。前段の入力のSequenceから作られ、後段の出力のSequenceを構成するのに利用される「固定次元のベクトル」とは、二つの文が「同じ意味」を持つことを表現している文の意味のベクトル表現に他なりません。

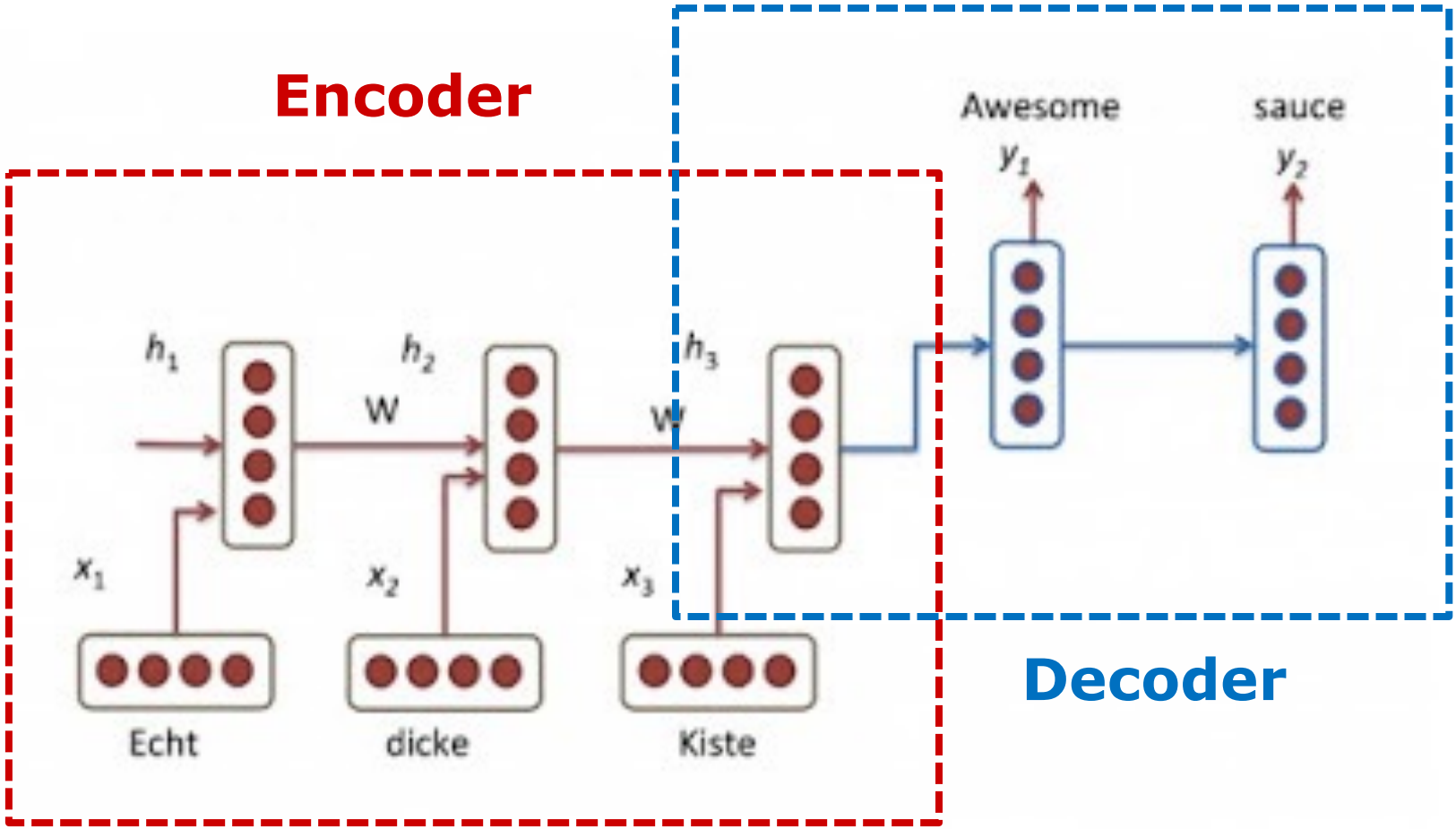
Sequence to Sequence

この論文の次の図を見て欲しい。

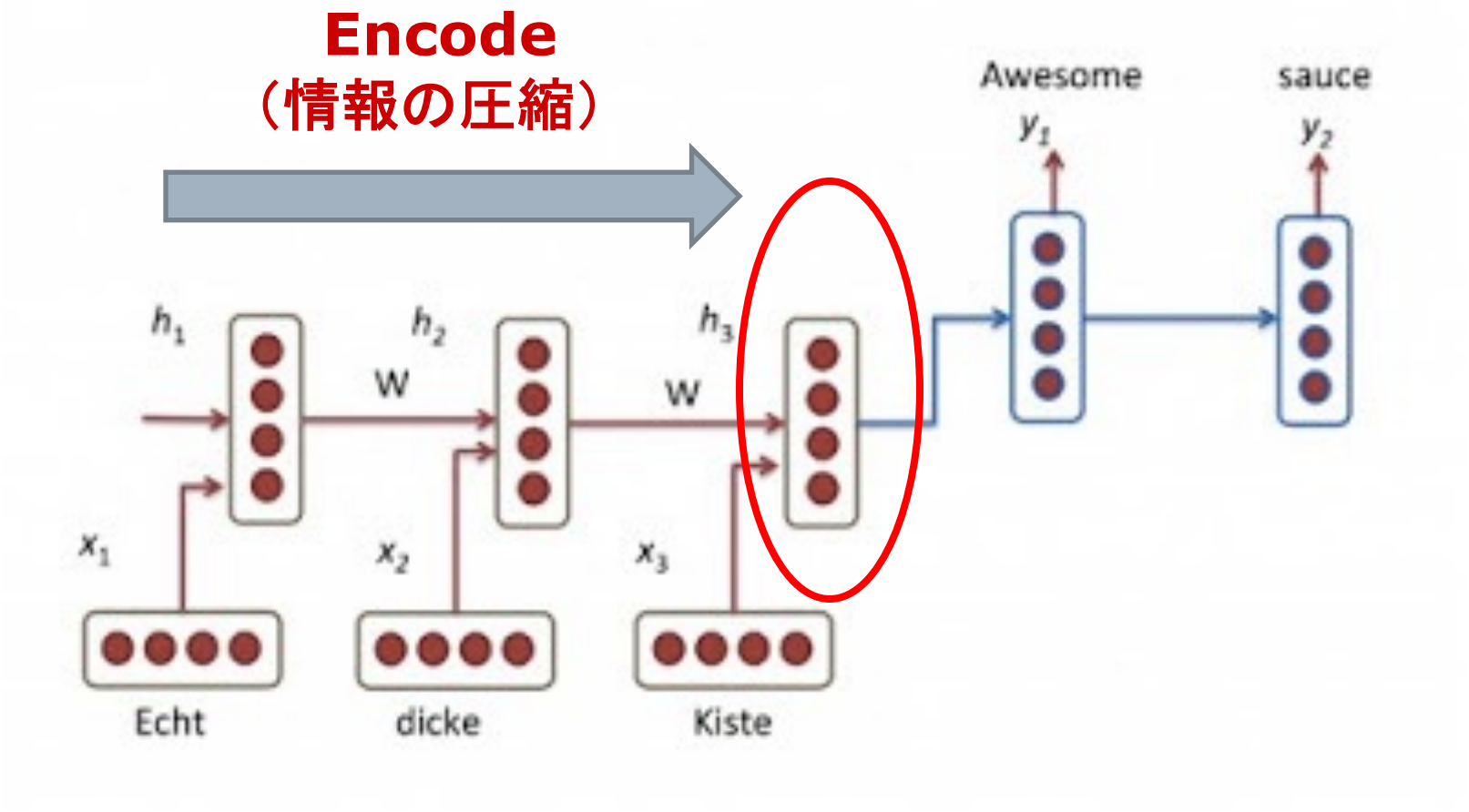


この図は、このシステムが、ABCというシーケンスが与えられた時、xyzというシーケンスを返すことを表している。<EOS>は、End of Sequence でシーケンスの終わりを表す特別な記号である。（これが、シーケンスの構造に課せられた「最小限の前提」である。）

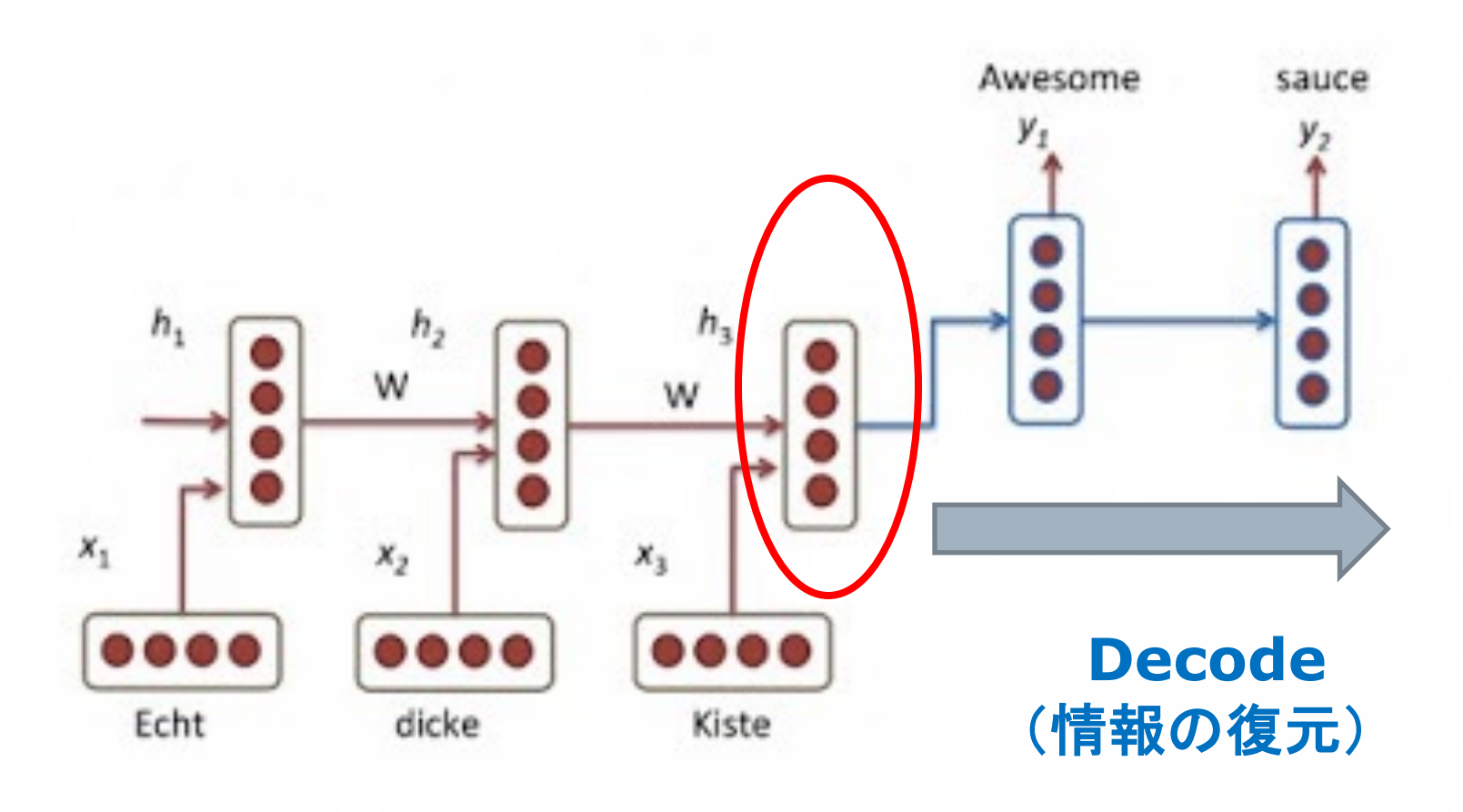
次の図(https://goo.gl/JGckBP から)は、こうしたメカニズムで、RNNが、独文の "Echt dicke Kiste" を英文の"Awesome sauce" に翻訳する様子を表している。(ここでは、文章の終わりを表す<EOS>は、省略されている)



ここでは、Encoder部が、文章の最後にとるRNNの内部状態 h_3 が、そのままDecoder部に渡されることが示されている。入力シーケンスの情報のエッセンスが、この内部状態 h_3に凝縮されていると考えればいい。



AutoencoderのDecoder部が、圧縮された情報から元の情報を復元しようとするように、ここでは、その情報から、「**同じ意味**」を持つ、別の言語の文章を復元しようとする。



Ilya Sutskever らは、このアーキテクチャーで、英語をフランス語に翻訳するシステムを作成し、BLEUのスコアで、34.81という高得点をたたき出した。

この時のシステムは、5段重ねのLSTMで構成され、それぞれが8,000次元の状態からなる384M個のパラメーターを持つものだった。

RNNの不思議な力
RNNは文法を理解している

2015年

The Unreasonable Effectiveness of Recurrent Neural Networks”

Andrej Karpathy

http://karpathy.github.io/2015/05/21/rnn-effectiveness/

2015年

2015年のChatGPT

RNNによる数学論文とソースコードの生成

Karpathyは、Stacks Projectの膨大な数学論文 https://zbmath.org/software/31299 をRNNに学習させ、数学論文（モドキ）を生成してみせた。

彼はまた、Linuxのソースコードを RNNに学習させて、Cプログラムのソースコード（モドキ）を生成してみせた。

For $\bigoplus_{n=1,\ldots,m}$ where $\mathcal{L}_{m\bullet} = 0$, hence we can find a closed subset $\mathcal{H}$ in $\mathcal{H}$ and any sets $\mathcal{F}$ on X, U is a closed immersion of S, then $U \to T$ is a separated algebraic space.

Proof. Proof of (1). It also start we get

$$S = \mathrm{Spec}(R) = U \times_X U \times_X U$$

and the compariciy in the fibre product covering we have to prove the lemma generated by $\coprod Z \times_U U \to V$. Consider the maps M along the set of points Sch_{fppf} and $U \to U$ is the fibre category of S in U in Section, ?? and the fact that any U affine, see Morphisms, Lemma ??. Hence we obtain a scheme S and any open subset $W \subset U$ in $Sh(G)$ such that $\mathrm{Spec}(R') \to S$ is smooth or an

$$U = \bigcup U_i \times_{S_i} U_i$$

which has a nonzero morphism we may assume that f_i is of finite presentation over S. We claim that $\mathcal{O}_{X,x}$ is a scheme where $x, x', s'' \in S'$ such that $\mathcal{O}_{X,x'} \to \mathcal{O}'_{X',x'}$ is separated. By Algebra, Lemma ?? we can define a map of complexes $\mathrm{GL}_{S'}(x'/S'')$ and we win. □

To prove study we see that $\mathcal{F}|_U$ is a covering of $\mathcal{X}'$, and $\mathcal{T}_i$ is an object of $\mathcal{F}_{X/S}$ for $i > 0$ and $\mathcal{F}_p$ exists and let $\mathcal{F}_i$ be a presheaf of $\mathcal{O}_X$-modules on $\mathcal{C}$ as a $\mathcal{F}$-module. In particular $\mathcal{F} = U/\mathcal{F}$ we have to show that

$$\widetilde{M}^\bullet = \mathcal{I}^\bullet \otimes_{\mathrm{Spec}(k)} \mathcal{O}_{S,s} - i_X^{-1}\mathcal{F})$$

is a unique morphism of algebraic stacks. Note that

$$\mathrm{Arrows} = (Sch/S)^{opp}_{fppf}, (Sch/S)_{fppf}$$

and

$$V = \Gamma(S, \mathcal{O}) \longmapsto (U, \mathrm{Spec}(A))$$

is an open subset of X. Thus U is affine. This is a continuous map of X is the inverse, the groupoid scheme S.

Proof. See discussion of sheaves of sets.

RNNが産み出した数学論文モドキ

Stack Theory の教科書を「学習」させたもの

Proof. Omitted. □

Lemma 0.1. *Let $\mathcal{C}$ be a set of the construction.*

Let $\mathcal{C}$ be a gerber covering. Let $\mathcal{F}$ be a quasi-coherent sheaves of $\mathcal{O}$-modules. We have to show that

$$\mathcal{O}_{\mathcal{O}_X} = \mathcal{O}_X(\mathcal{L})$$

.

Proof. This is an algebraic space with the composition of sheaves $\mathcal{F}$ on $X_{\acute{e}tale}$ we have

$$\mathcal{O}_X(\mathcal{F}) = \{morph_1 \times_{\mathcal{O}_X} (\mathcal{G}, \mathcal{F})\}$$

where $\mathcal{G}$ defines an isomorphism $\mathcal{F} \to \mathcal{F}$ of $\mathcal{O}$-modules. □

Lemma 0.2. *This is an integer $\mathcal{Z}$ is injective.*

Proof. See Spaces, Lemma ??. □

Lemma 0.3. *Let S be a scheme. Let X be a scheme and X is an affine open covering. Let $\mathcal{U} \subset \mathcal{X}$ be a canonical and locally of finite type. Let X be a scheme. Let X be a scheme which is equal to the formal complex.*

The following to the construction of the lemma follows.

Let X be a scheme. Let X be a scheme covering. Let

$$b : X \to Y' \to Y \to Y \to Y' \times_X Y \to X.$$

be a morphism of algebraic spaces over S and Y.

Proof. Let X be a nonzero scheme of X. Let X be an algebraic
quasi-coherent sheaf of $\mathcal{O}_X$-modules. The following are equivalent

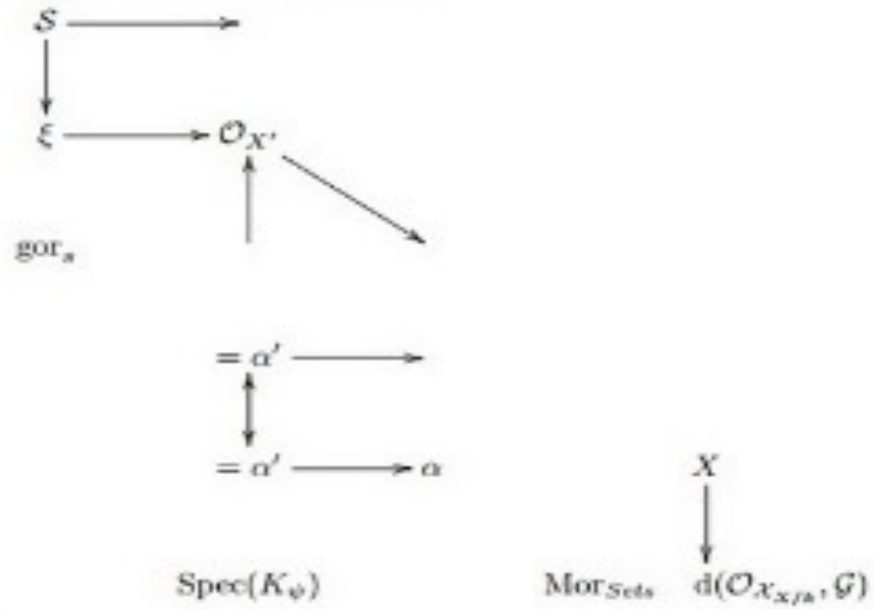
(1) $\mathcal{F}$ is an algebraic space over S.
(2) If X is an affine open covering.

Consider a common structure on X and X the functor $\mathcal{O}_X(U)$ v
finite type.

RNNが産み出した
数学論文モドキ

Stack Theory の教科書を
「学習」させたもの

This since $\mathcal{F} \in \mathcal{F}$ and $x \in \mathcal{G}$ the diagram



is a limit. Then $\mathcal{G}$ is a finite type and assume S is a flat and $\mathcal{F}$ and $\mathcal{G}$ is a finite type f_*. This is of finite type diagrams, and

- the composition of $\mathcal{G}$ is a regular sequence,
- $\mathcal{O}_{X'}$ is a sheaf of rings.

□

Proof. We have see that $X = \mathrm{Spec}(R)$ and $\mathcal{F}$ is a finite type representable by algebraic space. The property $\mathcal{F}$ is a finite morphism of algebraic stacks. Then the cohomology of X is an open neighbourhood of U.

Proof. This is clear that $\mathcal{G}$ is a finite presentation, see Lemmas ??.
A *reduced above* we conclude that U is an open covering of $\mathcal{C}$. The fu
"field

$$\mathcal{O}_{X,x} \longrightarrow \mathcal{F}_{\overline{x}} \quad \text{-1}(\mathcal{O}_{X_{\acute{e}tale}}) \longrightarrow \mathcal{O}_{X_i}^{-1}\mathcal{O}_{X_\lambda}(\mathcal{O}_{X_\nu}^{\overline{x}})$$

is an isomorphism of covering of $\mathcal{O}_{X_i}$. If $\mathcal{F}$ is the unique element of $\mathcal{F}$ such that X is an isomorphism.
The property $\mathcal{F}$ is a disjoint union of Proposition ?? and we can filtered set of presentations of a scheme $\mathcal{O}_X$-algebra with $\mathcal{F}$ are opens of finite type o
If $\mathcal{F}$ is a scheme theoretic image points.

If $\mathcal{F}$ is a finite direct sum $\mathcal{O}_{X_\lambda}$ is a closed immersion, see Lemma ??
sequence of $\mathcal{F}$ is a similar morphism.

RNNが産み出した
数学論文モドキ

Stack Theory の教科書を
「学習」させたもの

RNNは、Latexの構文規則を学習し、
こうしたシーケンスを構成できる。

```
\begin{proof}
We may assume that $\mathcal{I}$ is an abelian sheaf on $\mathcal{C}$.
\item Given a morphism $\Delta : \mathcal{F} \to \mathcal{I}$
is an injective and let $\mathfrak q$ be an abelian sheaf on $X$.
Let $\mathcal{F}$ be a fibered complex. Let $\mathcal{F}$ be a category.
\begin{enumerate}
\item \hyperref[setain-construction-phantom]{Lemma}
\label{lemma-characterize-quasi-finite}
Let $\mathcal{F}$ be an abelian quasi-coherent sheaf on $\mathcal{C}$.
Let $\mathcal{F}$ be a coherent $\mathcal{O}_X$-module. Then
$\mathcal{F}$ is an abelian catenary over $\mathcal{C}$.
\item The following are equivalent
\begin{enumerate}
\item $\mathcal{F}$ is an $\mathcal{O}_X$-module.
\end{lemma}
```

```
/*
 * Increment the size file of the new incorrect UI_FILTER group information
 * of the size generatively.
 */
static int indicate_policy(void)
{
 int error;
 if (fd == MARN_EPT) {
  /*
   * The kernel blank will coeld it to userspace.
   */
  if (ss->segment < mem_total)
   unblock_graph_and_set_blocked();
  else
   ret = 1;
  goto bail;
 }
 segaddr = in_SB(in.addr);
 selector = seg / 16;
 setup_works = true;
 for (i = 0; i < blocks; i++) {
  seq = buf[i++];
  bpf = bd->bd.next + i * search;
  if (fd) {
   current = blocked;
  }
```

RNNが産み出した
Cプログラム・モドキ

Linuxのソースコードを
「学習」させたもの

```
/*
 * If this error is set, we will need anything right after that BSD.
 */
static void action_new_function(struct s_stat_info *wb)
{
  unsigned long flags;
  int lel_idx_bit = e->edd, *sys & ~((unsigned long) *FIRST_COMPAT);
  buf[0] = 0xFFFFFFFF & (bit << 4);
  min(inc, slist->bytes);
  printk(KERN_WARNING "Memory allocated %02x/%02x, "
    "original MLL instead\n"),
    min(min(multi_run - s->len, max) * num_data_in),
    frame_pos, sz + first_seg);
  div_u64_w(val, inb_p);
  spin_unlock(&disk->queue_lock);
  mutex_unlock(&s->sock->mutex);
  mutex_unlock(&func->mutex);
  return disassemble(info->pending_bh);
}

static void num_serial_settings(struct tty_struct *tty)
{
  if (tty == tty)
    disable_single_st_p(dev);
  pci_disable_spool(port);
  return 0;
}
```

RNNが産み出した
Cプログラム・モドキ

Linuxのソースコードを
「学習」させたもの

これらの生成されたテキストは「意味」を欠いているのだが、....

生成された記事も数学論文もソースコードもは、意味をなさない。

これらの生成されたテキストは「意味」を欠いている。

ただし、「文法的」「構文的」には、正しいように見える。

RNNは、文法を理解できている!!


Part 2 大規模言語モデルへ

意味の分散表現論の系譜
Agenda

Part 2 大規模言語モデルへ

- 2016年：Attention Mechanism
- 2016年：Google ニューラル機械翻訳
- 2017年：Transformer
- 2019年：BERT

Attention Mechanism

Neural machine translation by jointly learning to align and translate

Bahdanau, D., Cho, K., and Bengio, Y

https://arxiv.org/pdf/1409.0473.pdf

2016年

論文の概要

近年、ニューラル機械翻訳として提案されたモデルは、多くの場合、Encoder-Decoderのファミリーに属している。そこでは、ソースの文が固定長ベクトルにエンコードされ、そこからデコーダが 翻訳文を生成する。

この論文では、固定長ベクトルの使用が、この基本的なEncoder/Decoderアーキテクチャの性能を改善する上でのボトルネックになっていると推論し、モデルに自動的に、ターゲット・ワードを予測するのに重要なソース・文の一部分について、（ソフト）検索を可能とすることによって、これを拡張することを提案する。

その際、これらの部分を明示的にハードセグメントとして形成する必要はない。

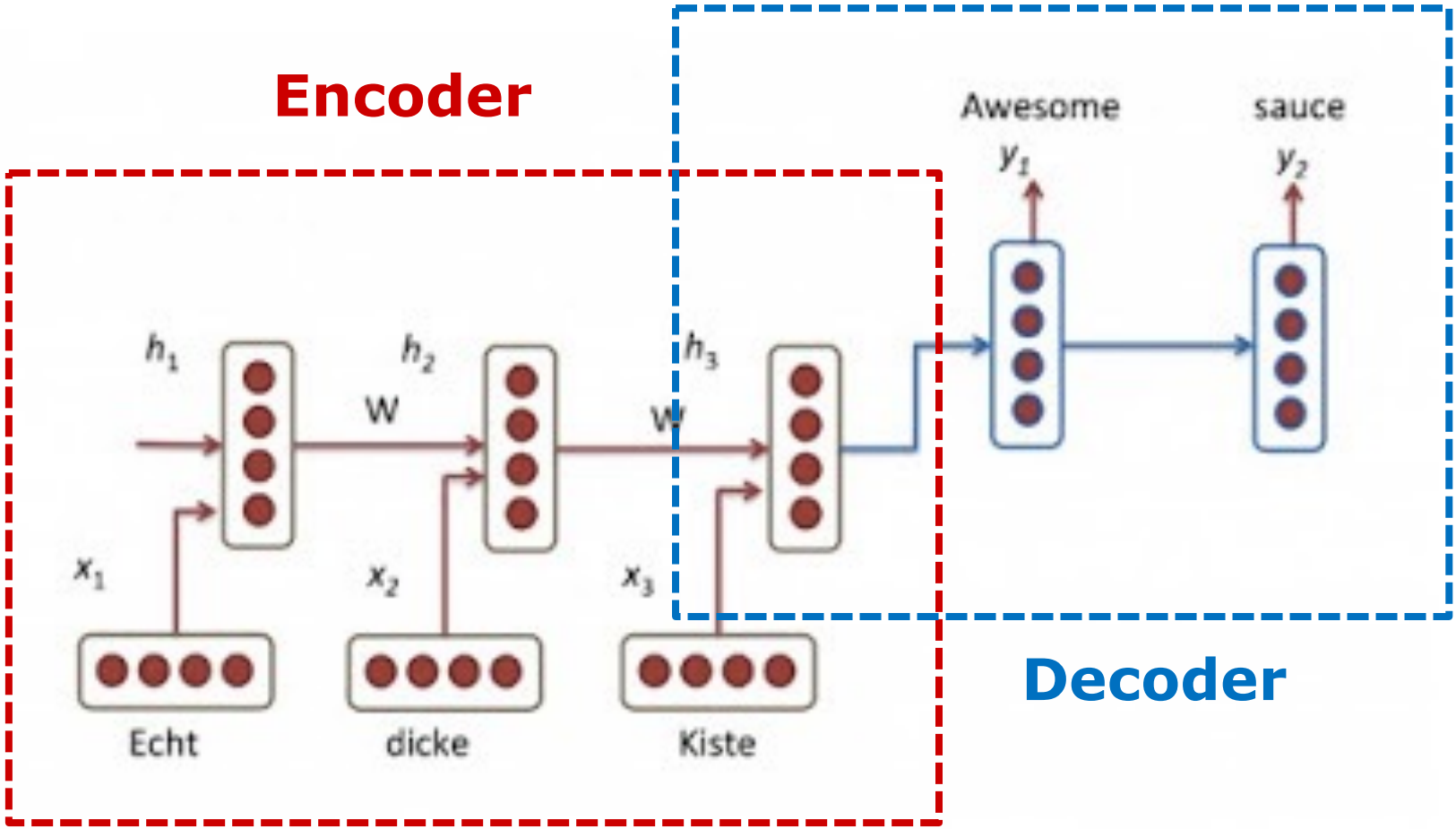
固定長ベクトルがボトルネック

先に見た、Ilya Sutskever らの翻訳システムでは、翻訳さるべき文は、Encoderで、一旦、ある決まった大きさの次元（例えば8000次元）を持つベクトルに変換される。このベクトルからDecoderが翻訳文を生成する。

入力された文が、長いものであっても短いものであっても、中間で生成され以降の翻訳プロセスすべての出発点となるこのベクトルの大きさは同じままだ。このシステムでは、長くても短くても入力された文全体が、一つの固定長のベクトルに変換されるのだ。

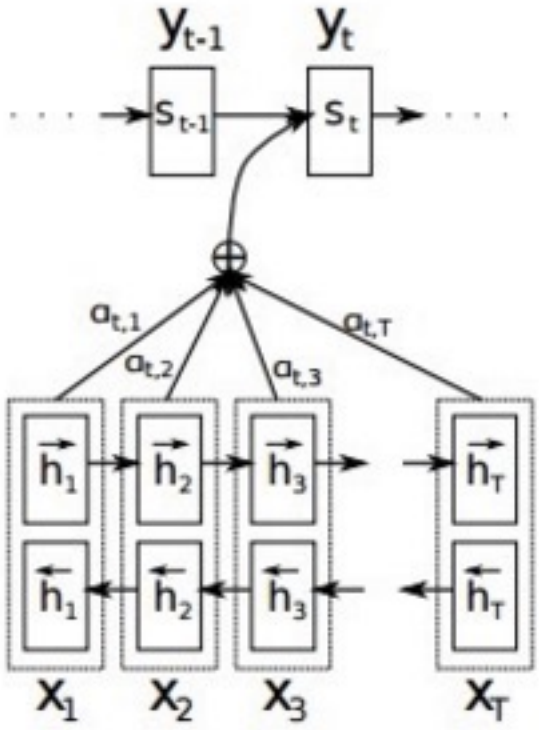
確かに、そこは翻訳の精度を上げる上でのボトルネックになりうる。事実、Ilya Sutskever らのシステムでは、文の長さが長くなるにつれて、翻訳の精度が低下されるのが観察されるという。

次の図(https://goo.gl/JGckBP から)は、こうしたメカニズムで、RNNが、独文の "Echt dicke Kiste" を英文の"Awesome sauce" に翻訳する様子を表している。(ここでは、文章の終わりを表す<EOS>は、省略されている)



この論文の基本的アイデア

文全体に一つの固定長のベクトルを割り当てるのではなく、翻訳時に、ソース・文の一部分を 改めて見直して、その部分から提供される情報を翻訳に生かそうということだ。



$a_{3,2}$が大きい場合、これは、Decorderがターゲット文の第3の単語を生成しながら、ソース文の第2の状態に多くの注意を払うことを意味する。

「ここで、yはデコーダによって生成された翻訳された単語であり、xは原文の単語である。上記の図は双方向のリカレント・ネットワークを使用しているが、それは重要ではない。逆方向は無視していい。

重要な部分は、各デコーダの出力するワード y_tが、Encoderの最後の状態だけでなく、すべての入力状態の重みづけられた結合に依存することである。

aは、出力ごとに、それぞれの入力状態をどの程度考慮されるべきかを定義する重みである。したがって、$a_{3,2}$が大きい場合、これは、Decorderがターゲット文の第3の単語を生成しながら、ソース文の第2の状態に多くの注意を払うことを意味する。

aは、通常、1に合計されるように正規化される（それらは、入力状態に対する確率分布である）。」

Google ニューラル機械翻訳

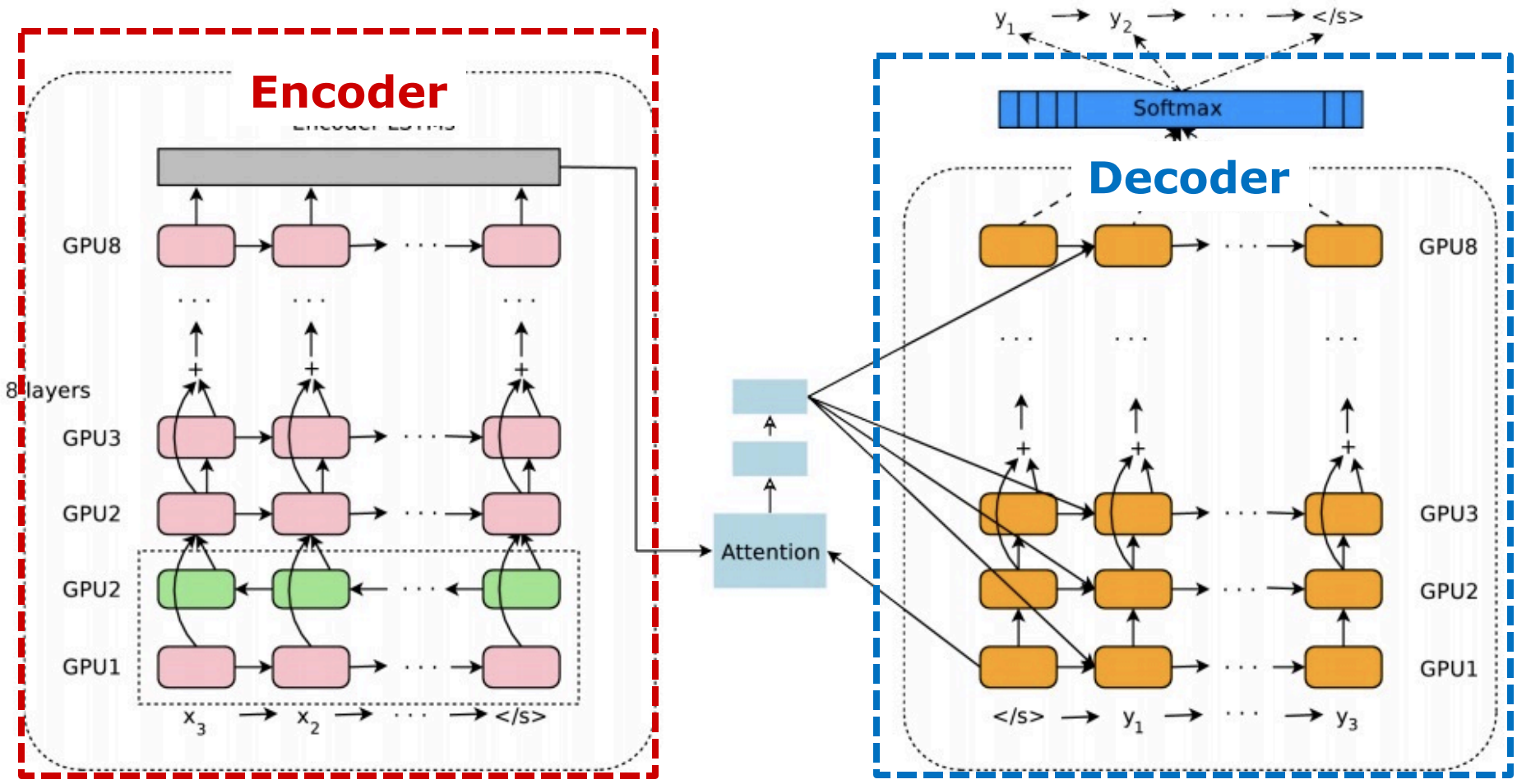
Google’s Neural Machine Translation System: Bridging the Gap between Human and Machine Translation

Yonghui Wu et al.

https://arxiv.org/pdf/1609.08144.pdf

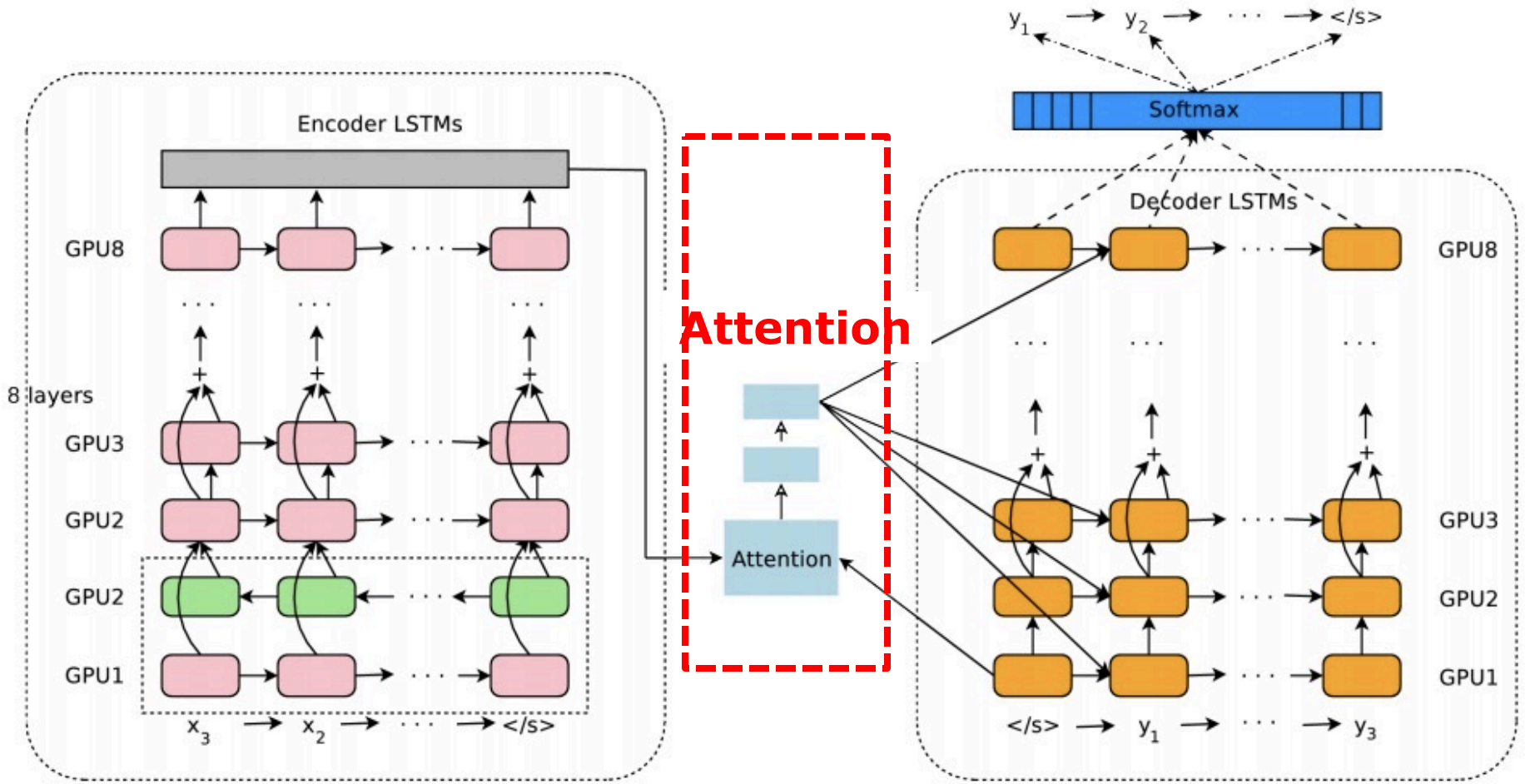
2016年

Encoder / Decoder



左側に、LSTMを8段重ねにした Encoder LSTMがあり、
右側には、同じくLSTMを8段重ねにした Decoder LSTMがある。

Attention Mechanism



EncoderとDecoderの中間に、Attentionと記された領域がある。
ここからの出力Attention Context は、Decoderのすべての
ノードに供給されている。

Google’s Multilingual Neural Machine Translation System: Enabling Zero-Shot Translation

Melvin Johnson et al.

https://arxiv.org/pdf/1611.04558.pdf

2016年

Google 多言語ニューラル機械翻訳

多言語の翻訳を、言語ペアの数だけのシステムによって行うのではなく、一つのシステムで行うことにより、さらに多くの言語へのスケール・アップが容易になる。システムが単純になり、コーパスの少ない言語にもメリットが生まれる。

ゼロ・ショット翻訳が可能であることを示し、また、インターリンガの存在を示唆するなど、非常に刺激的である。

言語に対するいくつかの「仮説」と、ディープラーニングでの結果をある種の「実験」として、結びつけようとするスタイルは新しいものである。

論文の概要

「我々は、単一のニューラル機械翻訳（NMT）モデルを使用して、複数の言語どうしを翻訳する、シンプルで洗練されたソリューションを提案する。」

「共有ワードピースのボキャブラリを使用することで、多言語NMTはパラメータを増やさずに、単一のモデルを利用することができる。これは、従来の多言語NMTの提案よりも大幅に簡単なものだ。」

「我々のモデルは、訓練中に明示的には見られなかった言語ペアの間の暗黙的な橋渡しを実行することも学ぶことができた。それは、翻訳の学習とゼロ・ショット翻訳がニューラル翻訳で可能であることを示している。」

「我々のモデルには、普遍的なインターリンガ表現が存在することを示唆する分析を示し、複数の言語を混在させた時に起きる、興味深い例を示す。」

ゼロ・ショット翻訳

このアプローチの興味深い利点は、明示的な訓練データがない言語ペアの間でゼロ・ショット翻訳を実行できることである。

これを実証するために、2つの異なる言語ペア、
ポルトガル語 -> 英語と英語 -> スペイン語（モデル1）と、
英語 <-> ポルトガル語、英語 <-> スペイン語（モデル2）の4つの異なる言語ペアのデータで訓練されたモデルの、2つの多言語モデルを使う。

これらのモデルの両方が、スペイン語 -> ポルトガル語で、合理的に良質な ポルトガル語を生成できることを示す。

著者らは、これを、「私たちの知る限り、これは真の多言語ゼロ・ショット翻訳の最初のデモです。」と主張している。

意味の共通表現の存在
インターリンガの存在

モデルを複数の言語にまたがってトレーニングすることで、個々の言語レベルでのパフォーマンスが向上し、ゼロ・ショット翻訳が有効になることがわかるということが、この論文の結論なのだが、その意味は？

言語にかかわらず、ネットワークは、同じ意味を持つ文章が同じような方法で表現される何らかの共有表現を学習しているのか？
この質問に対しては、そうだという。これも、画期的。

モデルは、訓練された言語ペアを扱うのと同じ方法で、ゼロ・ショット翻訳を実行しているのか？　これについては、まだよくわからないという。

インターリンガの証拠

いくつかの訓練されたネットワークは、実際に共有表現の強いビジュアルな証拠を示す。

たとえば、以下の図2は、英語 <-> JapaneseとEnglish <-> Koreanでトレーニングされた多対多モデルから作成されたものである。モデルの実際の動作を視覚化するために、意味論的に同一の言語間フレーズの74個のトリプルからなる小さなコーパスから始めた。

つまり、それぞれのトリプルには、同じ基本的な意味を持つ英語、日本語、韓国語のフレーズが含まれている。これらのトリプルをコンパイルするために、私たちは日本語と韓国語の翻訳と対になった、英語の文章のための正しい（Ground Truth）データベースを検索した。

英語

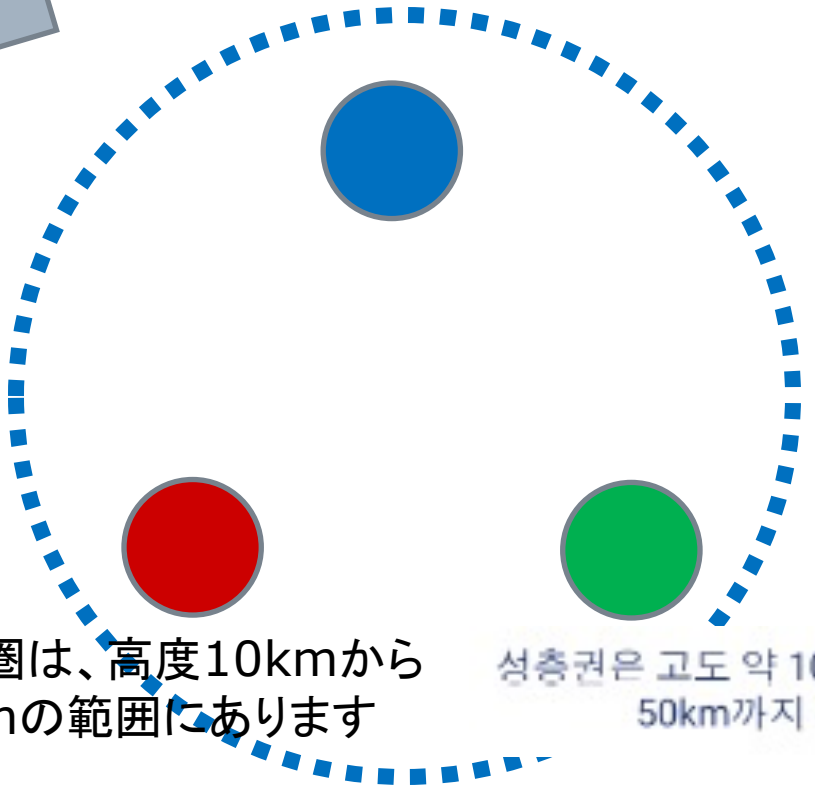
三つの言語で
同じ意味を持つ文を
一つのトリプルに
まとめる。

日本語

韓国語

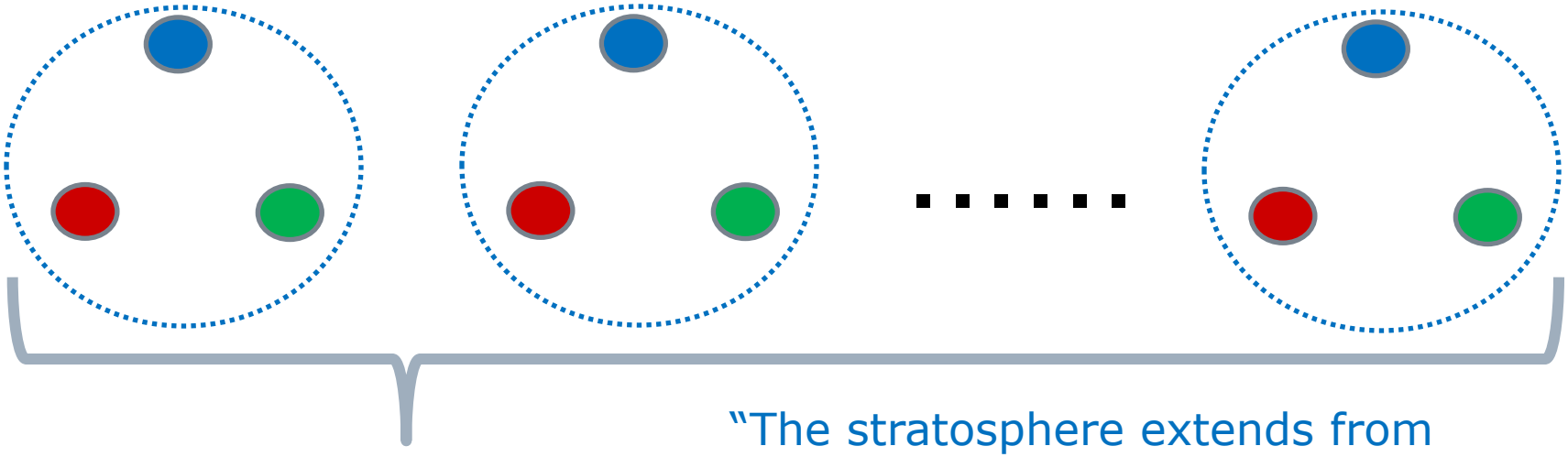
例えば

“The stratosphere extends from about 10km to about 50km in altitude.”



成層圏は、高度10kmから
50kmの範囲にあります

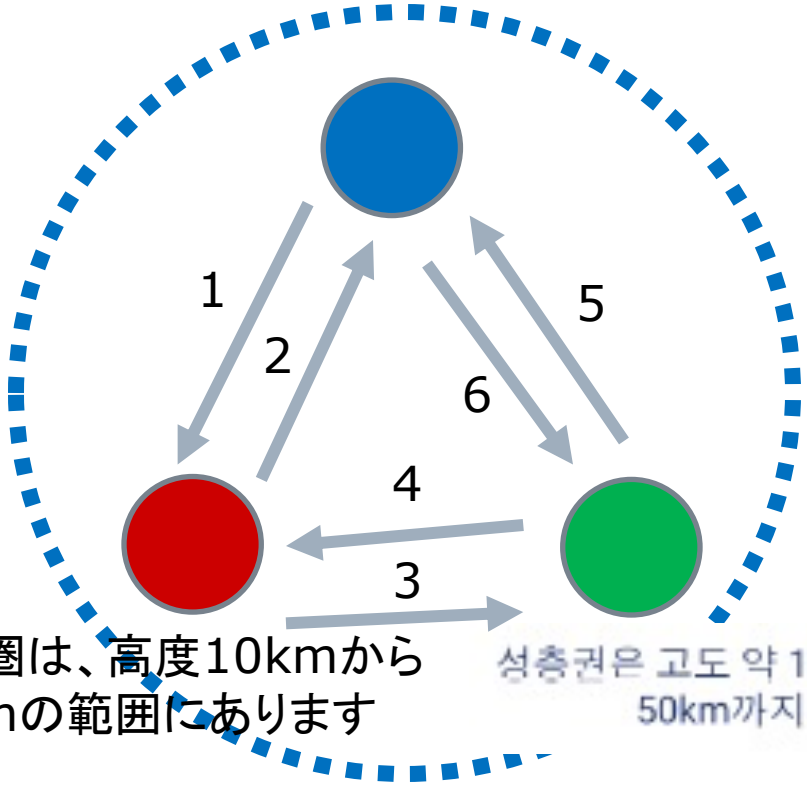
성층권은 고도 약 10km부터 약 50km까지 확장됩니다.



74個のトリプルからなる
コーパスを準備する。

"The stratosphere extends from about 10km to about 50km in altitude."

各トリプルの各センテンスを
他の2つの言語に翻訳する。
一つのトリプルについて、
6つの翻訳が生まれる。



このコーパスでは、
74 x 6 = 444 の
翻訳が生まれる。

成層圏は、高度10kmから
50kmの範囲にあります

言語にかかわらず、同様の意味を持つ文章がクラスタリングされている。

この 444の翻訳を行うのに、システムは9,978ステップ要した。

この時、発せられた9,978個のアテンションベクトルを、t-SNEを用いて二次元に投射したのが左の図。

同じトリプル内の文の翻訳で発せられるアテンション・ベクトルは、同じ色で表している。

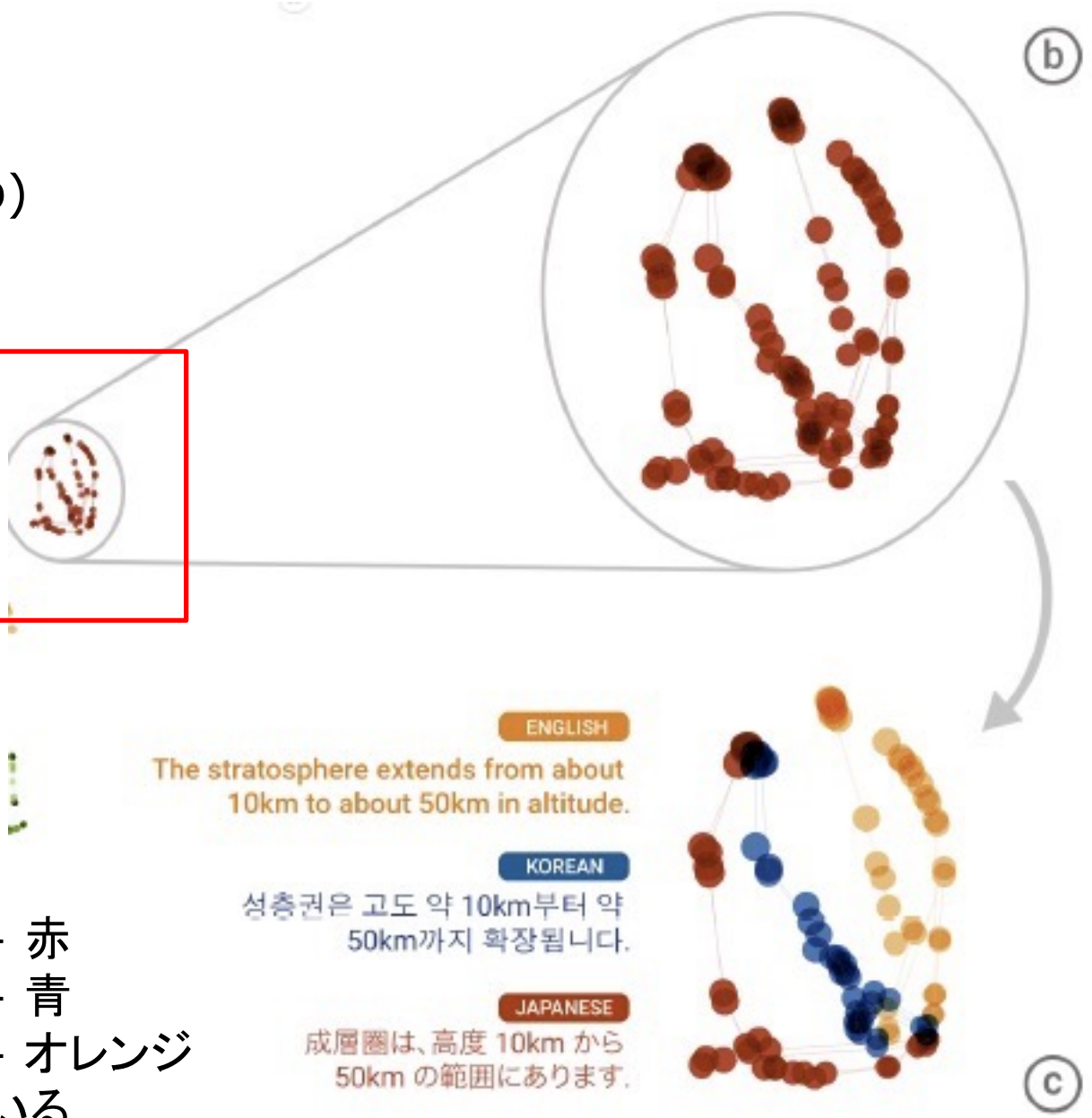
同じ色が、近くに集まってクラスターを形成しているのが見て取れる。（？）

この同じ色のクラスター(b)は、元の同じ意味を持つトリプルから生まれたものである。

同一の文から発せられるアテンション・ベクトルは、線で結ばれている。

(c)は、翻訳ソースが同じ言語を同じ色で塗り分けたもの。

日本語->{英語,韓国語} 赤
韓国語->{英語,日本語} 青
英語->{日本語,韓国語} オレンジ
は、クラスターを形成している。



意味の「同一性」 / 意味の共通表現の存在

- 言語が異なっていても、「同じ」意味を持つ語が存在する。
- 言語が異なっていても、「同じ」意味を持つ文が存在する。

機械翻訳技術から派生した大規模言語モデルの成功は、この「意味の共通表現」が存在し、それを機械が発見する能力を持つことによるものだと、考えることができる。

Transformer

Transformer – 大規模言語モデルの基礎

Transformer は、2017年にGoogleが発表したアーキテクチャーです。

それは、GoogleのBERTやOpenAIのGPTといった現代の大規模言語モデルほとんど全ての基礎になっています。BERTの最後の文字 'T' も、GPTの'T'も"Transformer" アーキテクチャーを採用していることを表しています。

Attentionこそすべて

Googleの論文のタイトルは、"Attention is all you need" というものでした。

「Attentionこそすべて」ということで、Attention Mechanism さえあれば、RNNもCNNもいらないと思い切ったことをいっています。

これは、大規模言語モデル成立への大きな飛躍だったと思います。

Attention Is All You Need

Ashish Vaswani et al.

https://papers.nips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf

2017年

Google ニューラル機械翻訳から継承したもの

まず最初に確認したいのは、見かけはずいぶん違って見えますが、Transformer アーキテクチャーは、大きな成功を収めた2016年のGoogle ニューラル機械翻訳のアーキテクチャーから多くを学んでいるということです。

ポイントをあげれば、Encode-Decoder アーキテクチャーの採用、EncoderとDecoderの分離、両者をつなぐAttention Mechanismの採用、等々。

こうした、Google ニューラル機械翻訳のアーキテクチャーの特徴は、そのまま、Transformerのアーキテクチャーに引き継がれています。それらの中で、Attentionこそが一番重要なのだというのが、Vaswani らの分析なのだと思います。

Attentionのパフォーマンスの向上
Multi-Head Attention

Attention Mechanismに注目すると、そこには特にパフォーマンスの面で課題があることも浮かび上がってきます。それは、シーケンシャルに実行され並列化されていません。また、Attentionが注目する二つの点の距離が離れると離れるほど計算量が増大します。

"Attention is all you need" 論文は、Attention Mechanismのパフォーマンス改善の提案なのです。"Multi Head Attention" は、まさに、そうしたものです。

もう一つ、この論文の大事な提案は、Self Attentionの重要性の指摘です。

Abstract

The dominant sequence transduction models are based on complex recurrent or convolutional neural networks that include an encoder and a decoder.

The best performing models also connect the encoder and decoder through an attention mechanism.

We propose a new simple network architecture, the Transformer, based solely on attention mechanisms, dispensing with recurrence and convolutions entirely.

Background

The goal of reducing sequential computation also forms the foundation of the Extended Neural GPU [20], ByteNet [15] and ConvS2S [8], all of which use convolutional neural networks as basic building block, computing hidden representations in parallel for all input and output positions.

In these models, the number of operations required to relate signals from two arbitrary input or output positions grows in the distance between positions, linearly for ConvS2S and logarithmically for ByteNet. This makes it more difficult to learn dependencies between distant positions [11].

In the Transformer this is reduced to a constant number of operations, albeit at the cost of reduced effective resolution due to averaging attention-weighted positions, an effect we counteract with Multi-Head Attention as described in section 3.2.

Self-attention, sometimes called intra-attention is an attention mechanism relating different positions of a single sequence in order to compute a representation of the sequence.

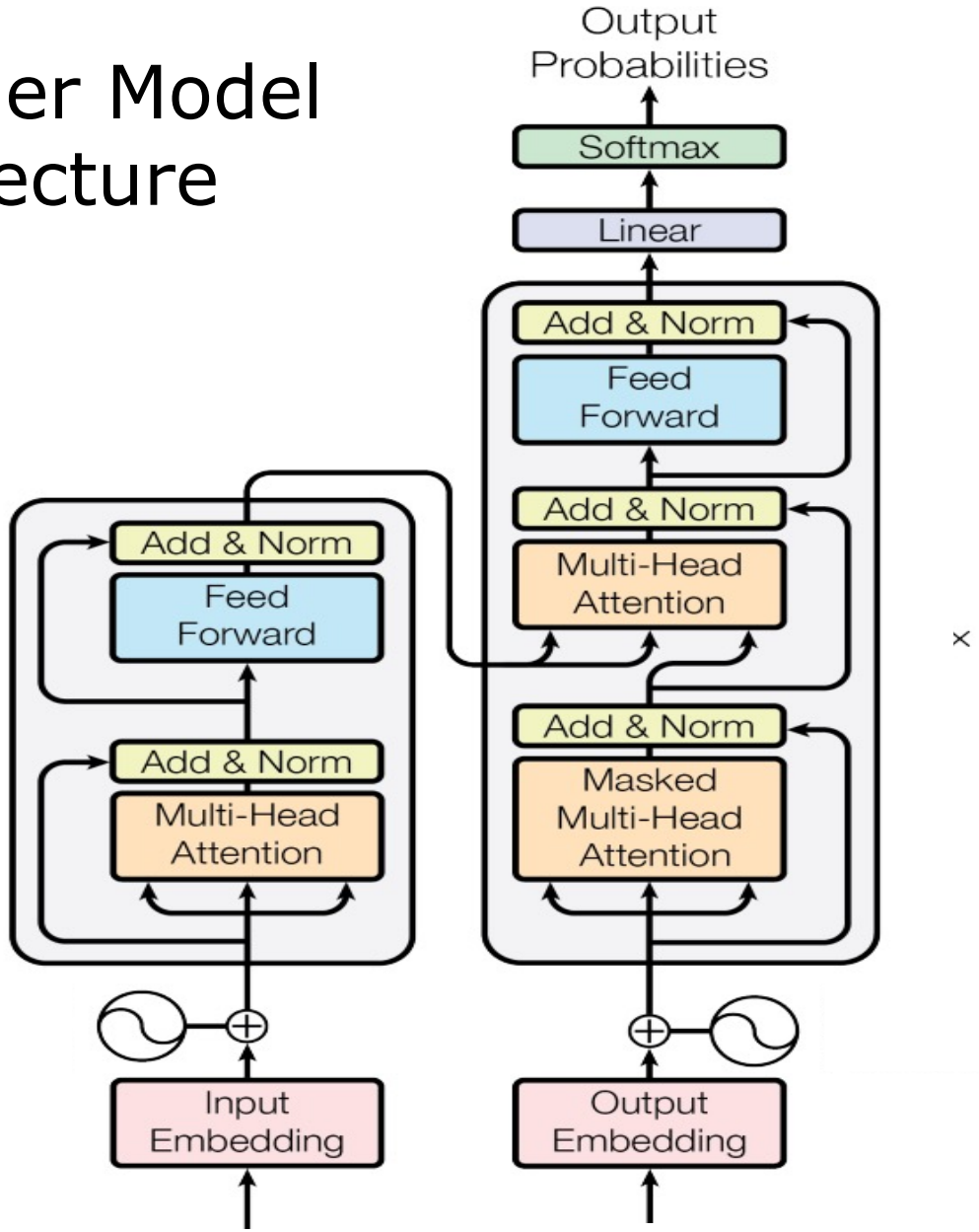
Self-attention has been used successfully in a variety of tasks including reading comprehension, abstractive summarization, textual entailment and learning task-independent sentence representations [4, 22, 23, 19].

End-to-end memory networks are based on a recurrent attention mechanism instead of sequence aligned recurrence and have been shown to perform well on simple-language question answering and language modeling tasks [28].

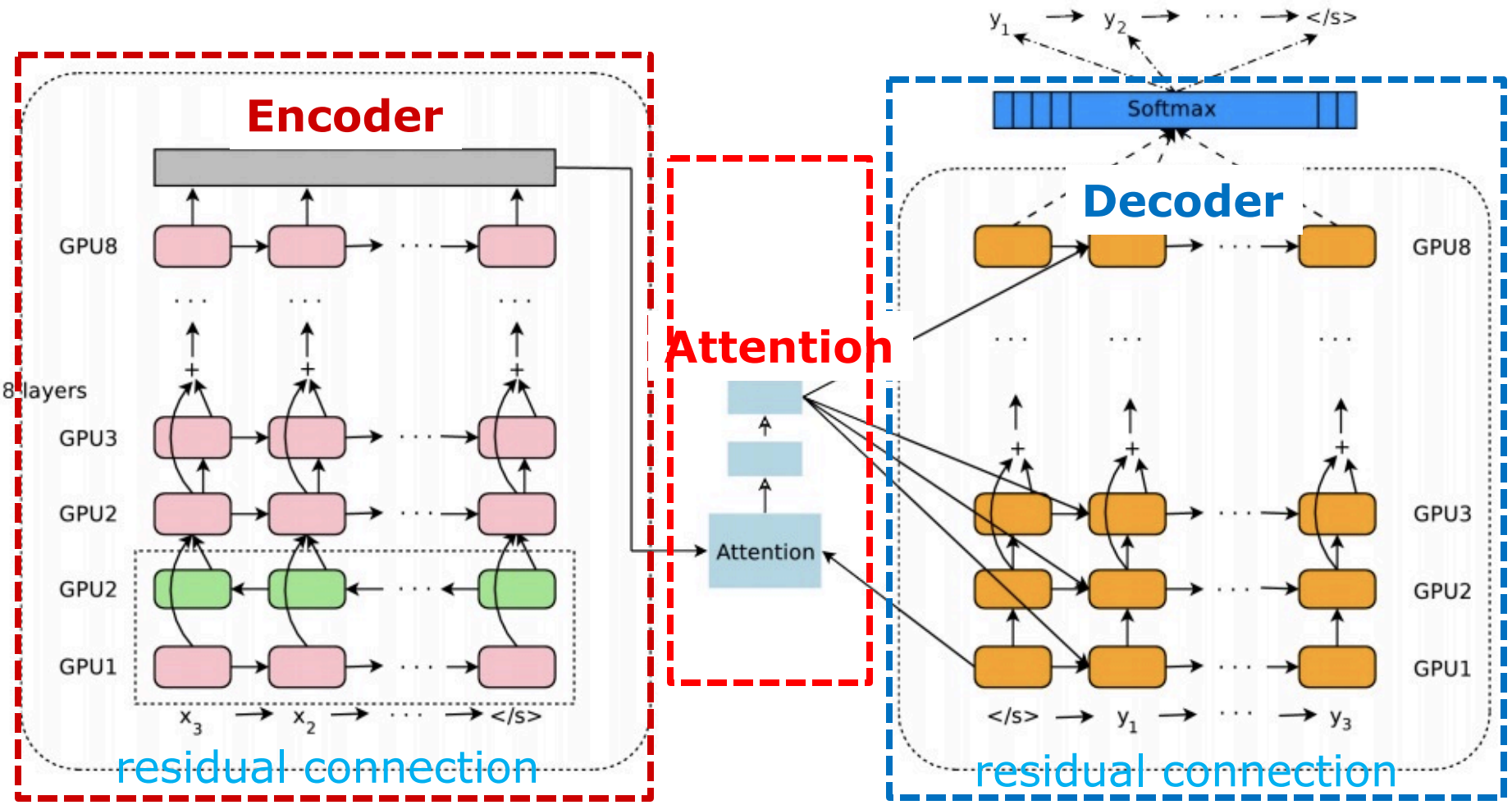
To the best of our knowledge, however, the Transformer is the first transduction model relying entirely on self-attention to compute representations of its input and output without using sequence aligned RNNs or convolution.

In the following sections, we will describe the Transformer, motivate self-attention and discuss its advantages over models such as [14, 15] and [8].

Transformer Model Architecture

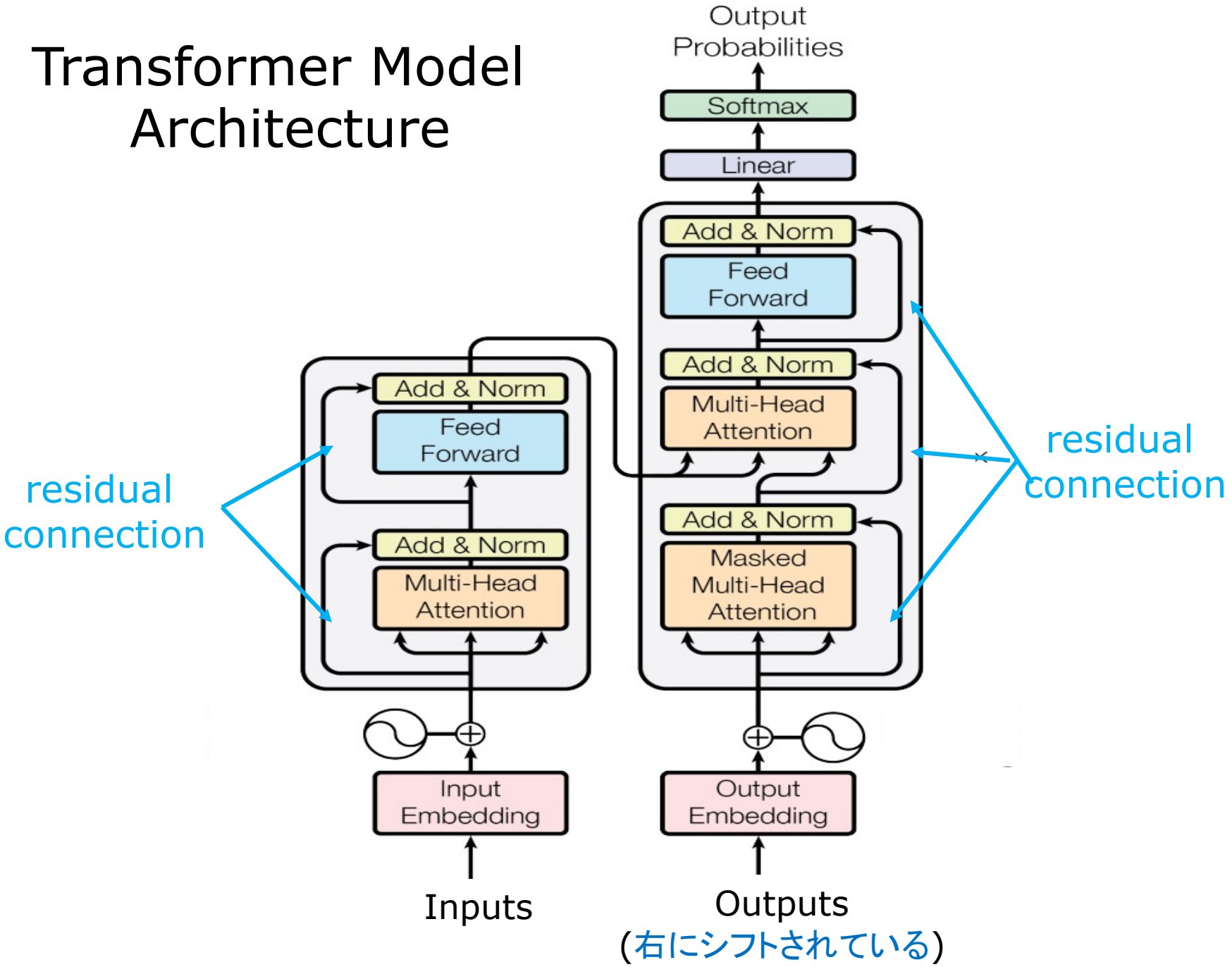


Google ニューラル機械翻訳のアーキテクチャー

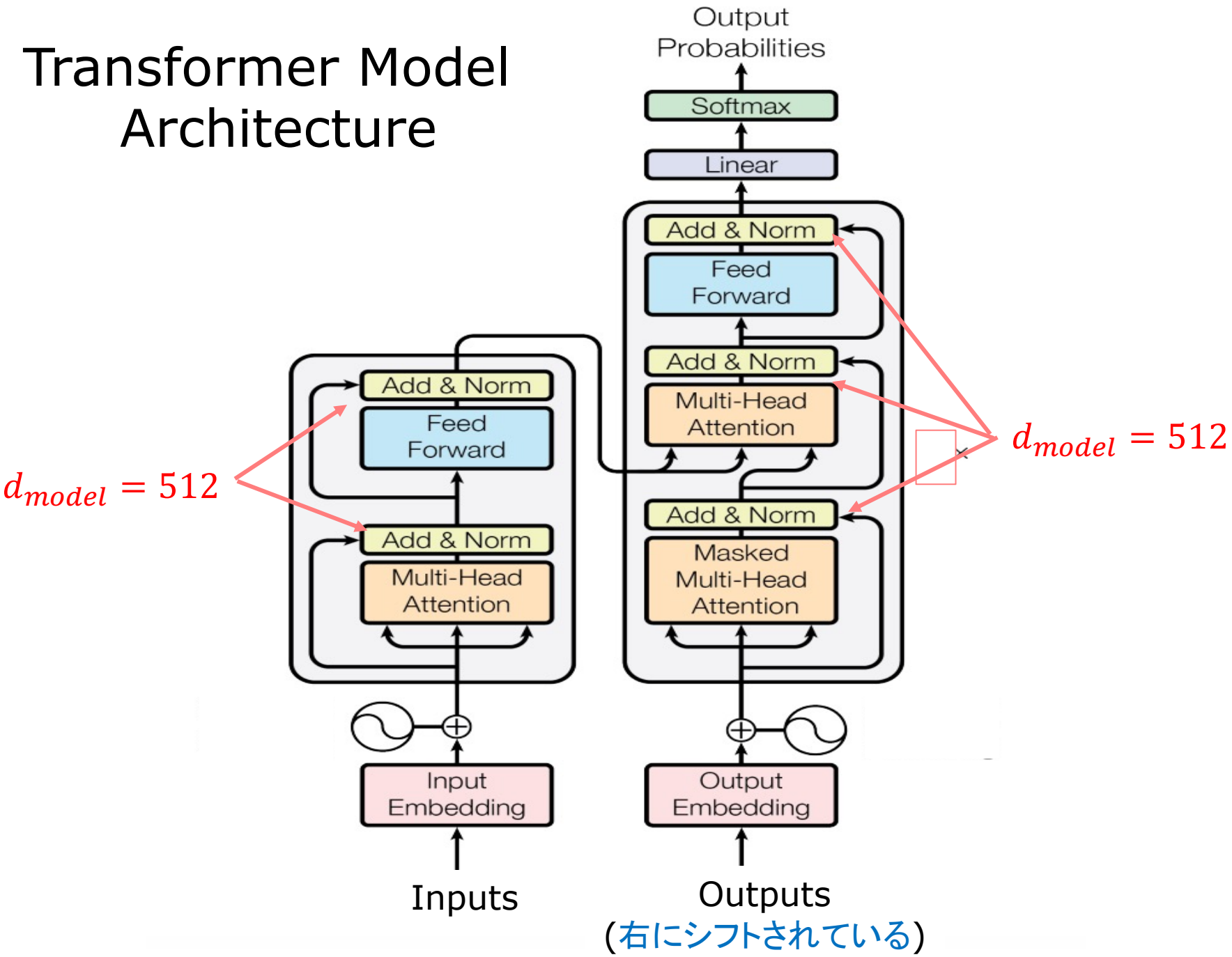


Encoder / Decoder/Attention

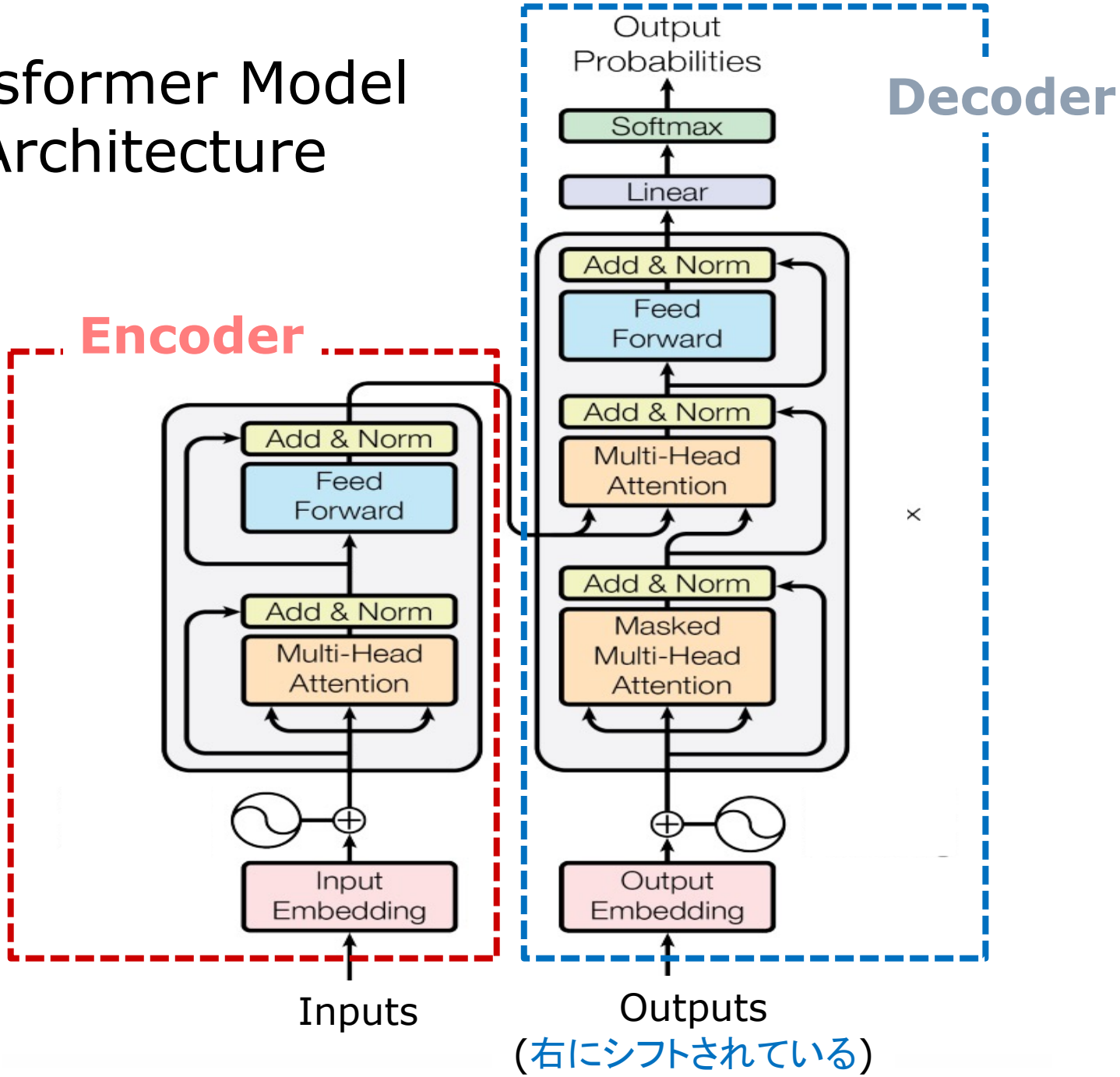
Transformer Model Architecture



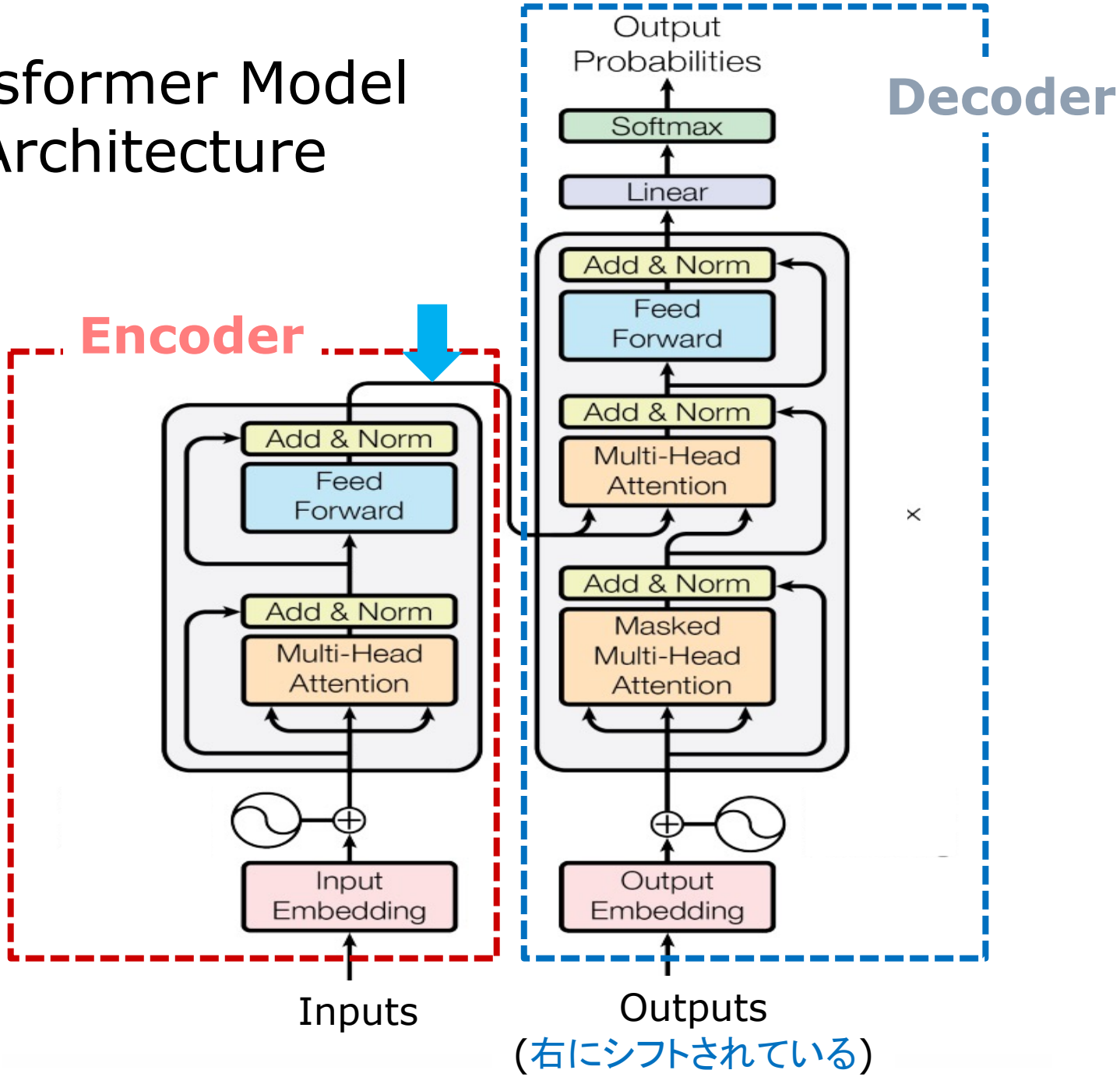
Transformer Model Architecture



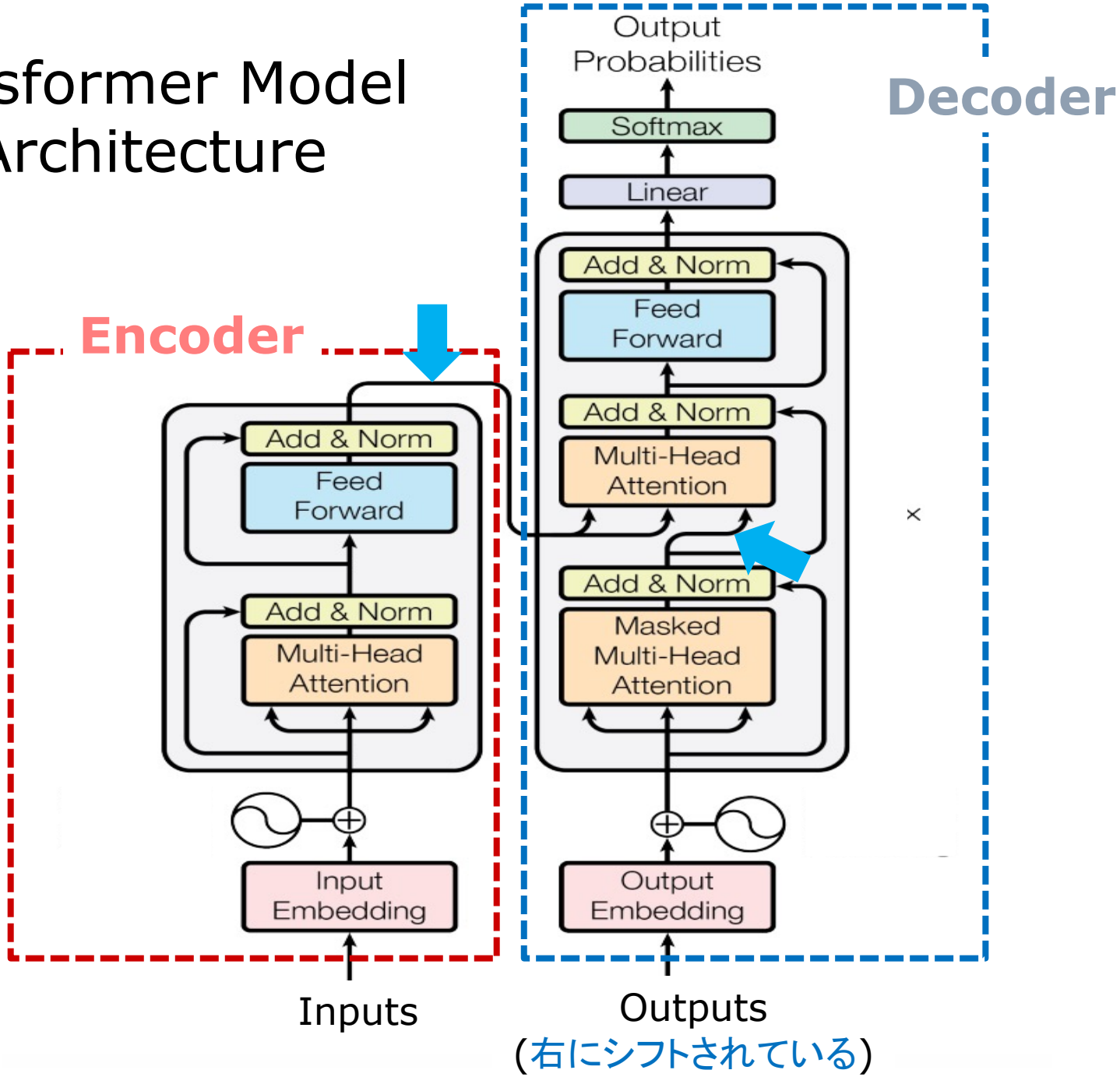
Transformer Model Architecture



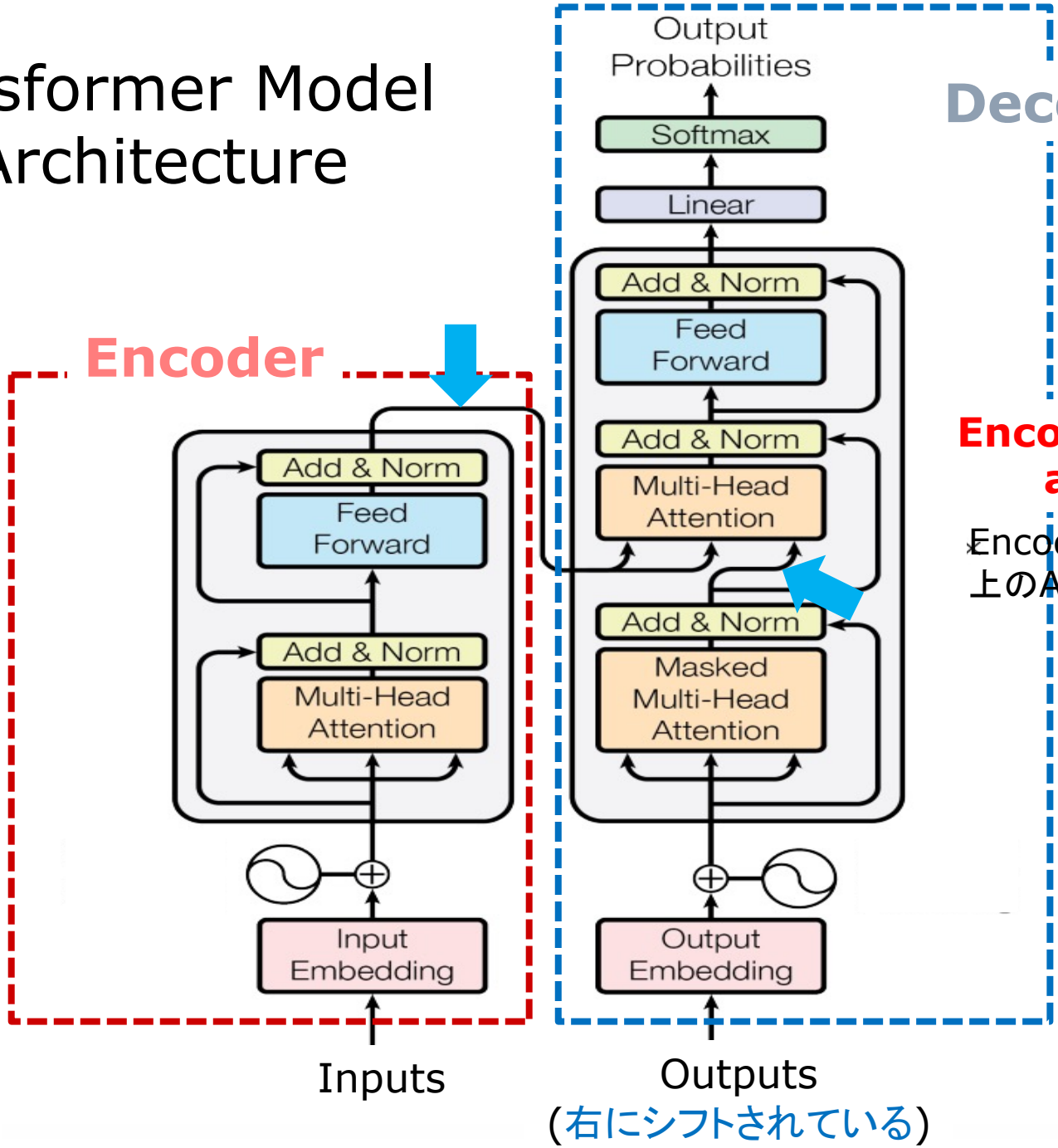
Transformer Model Architecture



Transformer Model Architecture



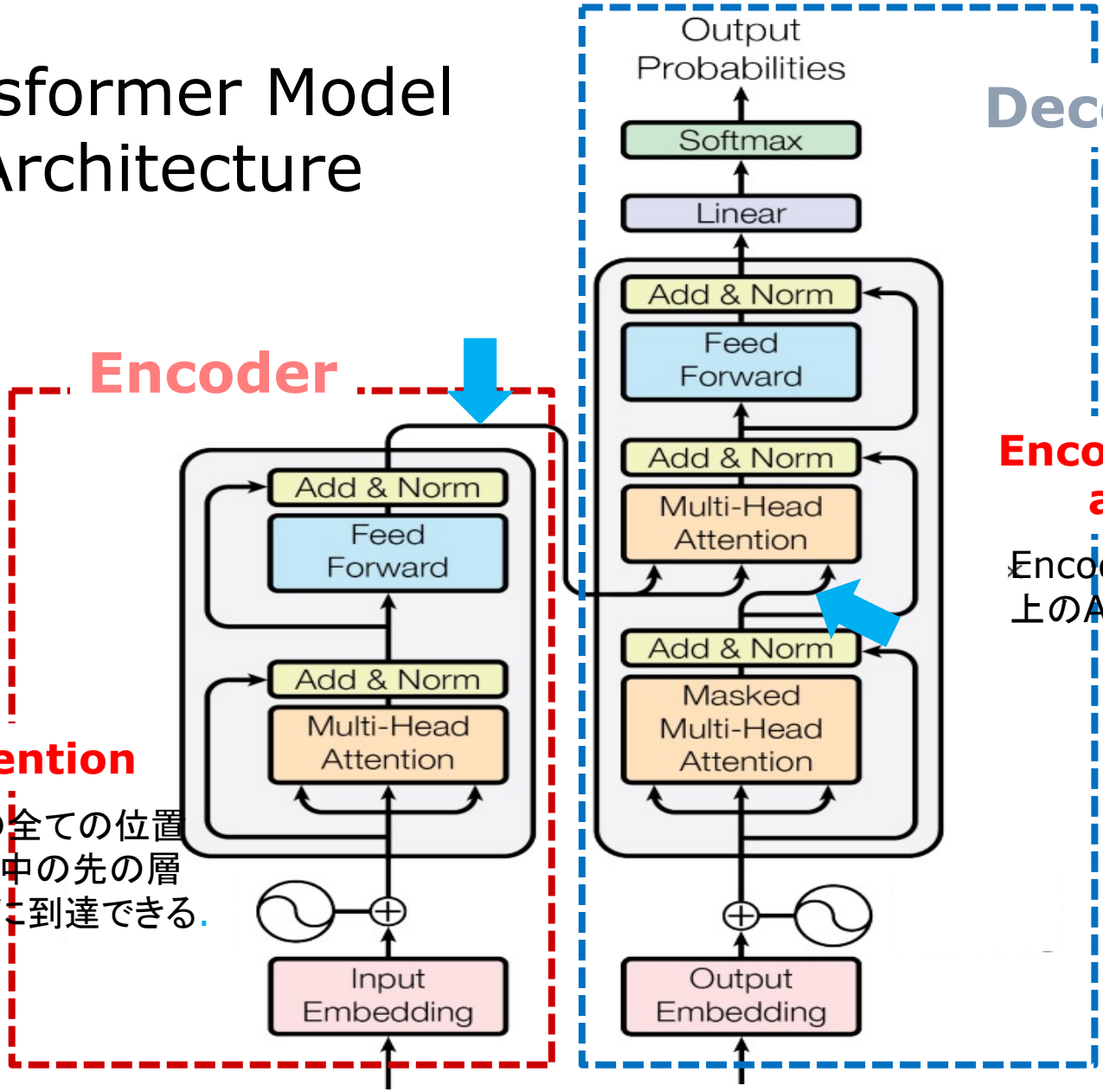
Transformer Model Architecture



Encoder-decoder attention

Encoderスタックの出力上のAttention

Transformer Model Architecture



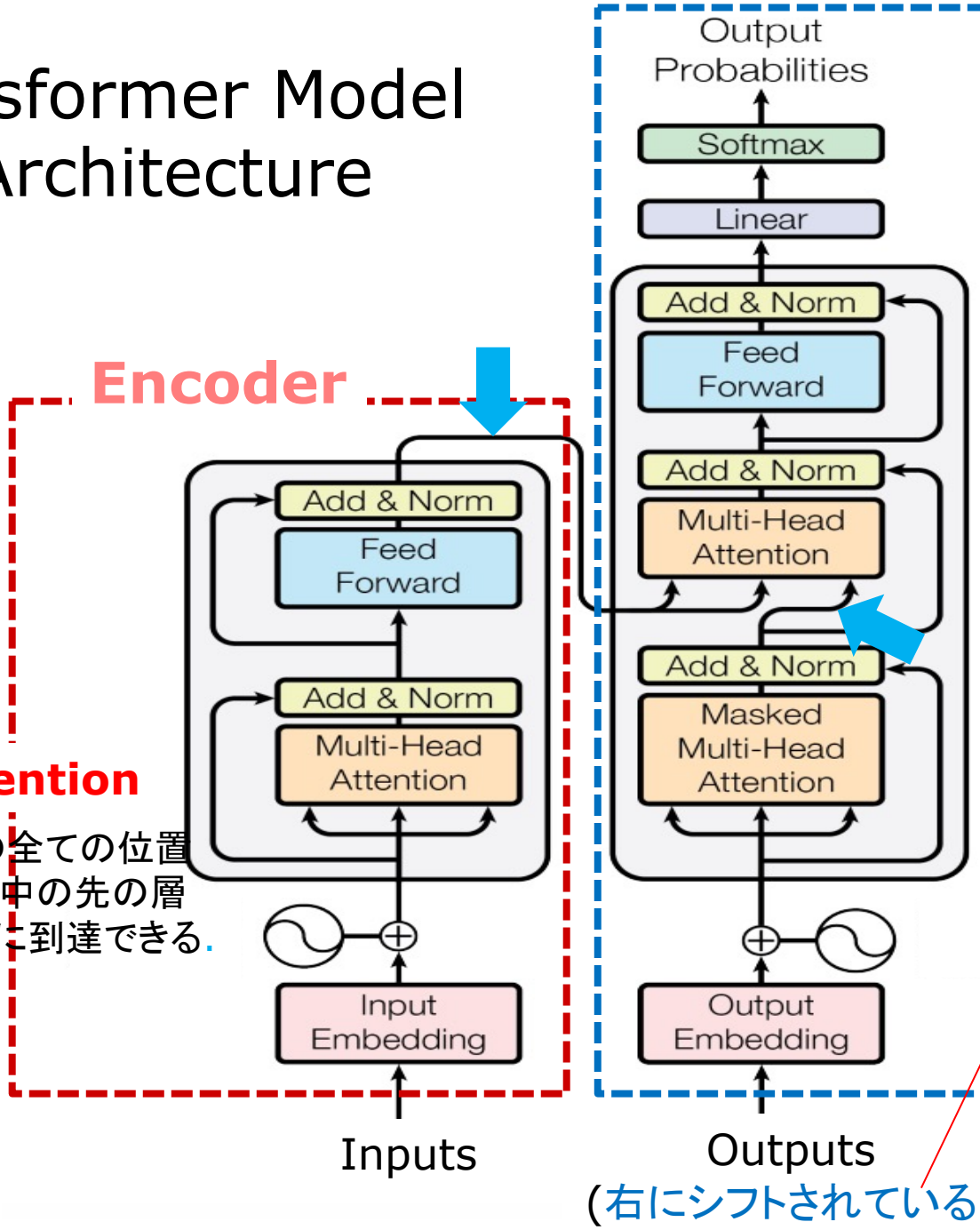
Self attention

Encoder中の全ての位置は、encoder中の先の層の全ての位置に到達できる.

Encoder-decoder attention

Encoderスタックの出力上のAttention

Transformer Model Architecture



Encoder-decoder attention

Encoderスタックの出力上のAttention

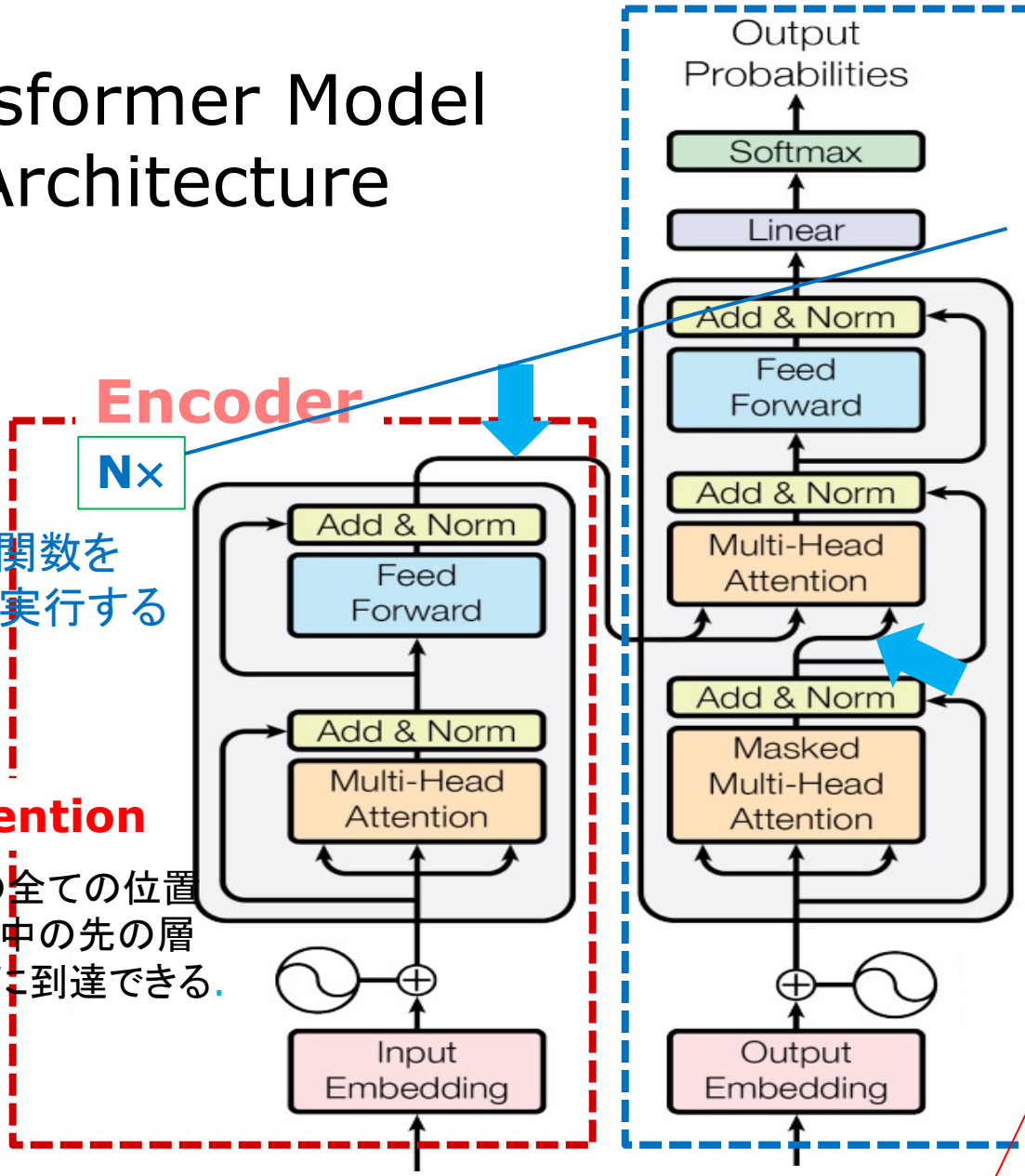
Masked Self attention

Decoderの中で、情報が肥大方向に流れるのを止める必要がある。

Self attention

Encoder中の全ての位置は、encoder中の先の層の全ての位置に到達できる.

Transformer Model Architecture



Encoder

N×

Attention関数を
N個並列に実行する

Self attention

Encoder中の全ての位置は、encoder中の先の層の全ての位置に到達できる.

Decoder

N×

Attention関数を
N個並列に実行する

Encoder-decoder attention

Encoderスタックの出力上のAttention

Masked Self attention

Decoderの中で、情報が左方向に流れるのを止める必要がある。

(右にシフトされている)

Model Architecture
Encoder and Decoder Stacks

Encoder: The encoder is composed of a stack of N = 6 identical layers. Each layer has two sub-layers. The first is a multi-head self-attention mechanism, and the second is a simple, position wise fully connected feed-forward network. We employ a residual connection [10] around each of the two sub-layers, followed by layer normalization [1]. That is, the output of each sub-layer is LayerNorm(x + Sublayer(x)), where Sublayer(x) is the function implemented by the sub-layer itself. To facilitate these residual connections, all sub-layers in the model, as well as the embedding layers, produce outputs of dimension dmodel = 512.

Model Architecture
Encoder and Decoder Stacks

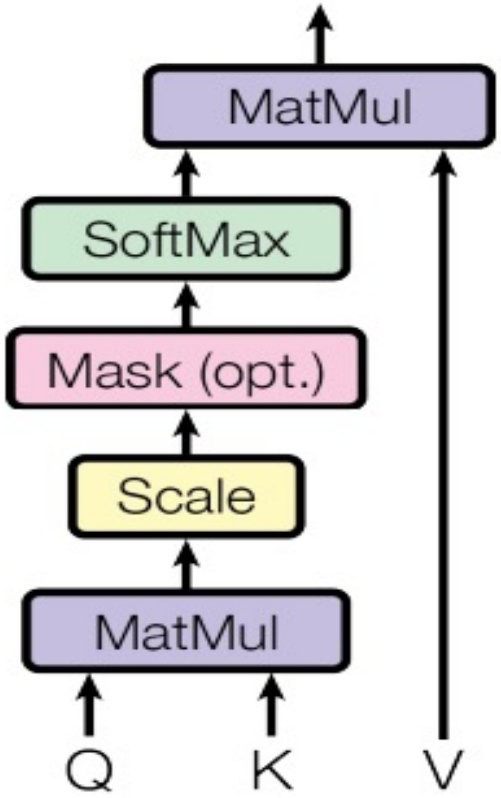
Decoder: The decoder is also composed of a stack of N = 6 identical layers. In addition to the two sub-layers in each encoder layer, the decoder inserts a third sub-layer, which performs multi-head attention over the output of the encoder stack. Similar to the encoder, we employ residual connections around each of the sub-layers, followed by layer normalization. We also modify the self-attention sub-layer in the decoder stack to prevent positions from attending to subsequent positions. This masking, combined with fact that the output embeddings are offset by one position, ensures that the predictions for position i can depend only on the known outputs at positions less than i.

Scaled Dot-Product Attention

We call our particular attention "Scaled Dot-Product Attention" (Figure 2). The input consists of queries and keys of dimension d_k, and values of dimension d_v. We compute the dot products of the query with all keys, divide each by $\sqrt{d_k}$, and apply a softmax function to obtain the weights on the values.

$$Attention(Q, K, V) = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

Scaled Dot-Product Attention



Queries **Keys** **Values**

$$Q \in \mathbb{R}^{d_k}$$
$$K \in \mathbb{R}^{d_k}$$
$$V \in \mathbb{R}^{d_v}$$

$$Attention(Q, K, V) = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

Multi-Head Attention

Instead of performing a single attention function with d_{model}-dimensional keys, values and queries,

we found it beneficial to linearly project the queries, keys and values h times with different, learned

linear projections to $d_k, d_k \; and \; d_v$ dimensions, respectively.

On each of these projected versions of queries, keys and values we then perform the attention function in parallel, yielding d_v-dimensional output values. These are concatenated and once again projected, resulting in the final values,

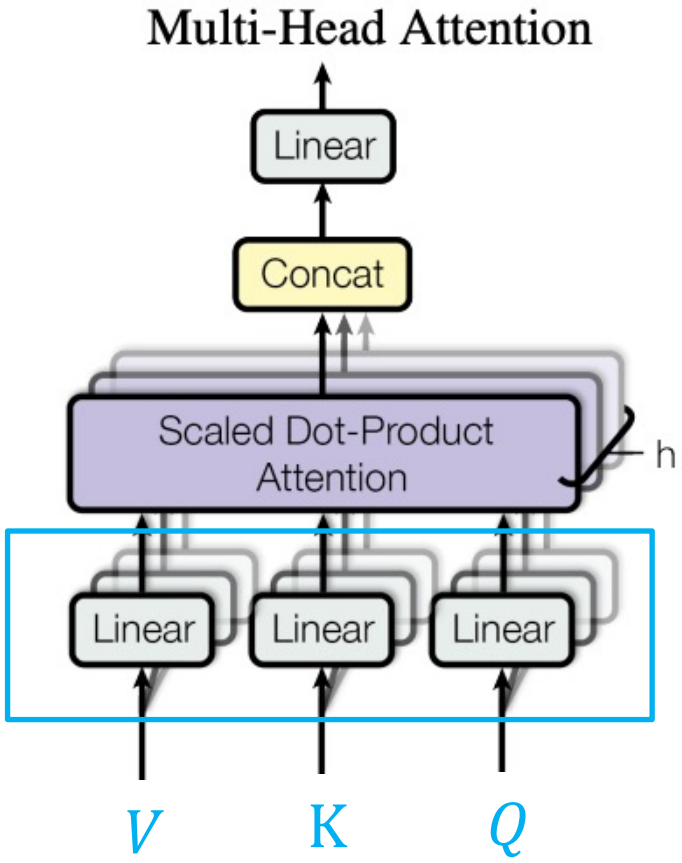
Multi-Head Attention

Multi-head attention allows the model to jointly attend to information from different representation subspaces at different positions. With a single attention head, averaging inhibits this.

$$MultiHead(Q, K, V) = concat(head_1, \cdots, head_h)W^O$$
$$where\ head_i = Attention(QW_i^Q, KW_i^K, VW_i^V)$$

Where the projections are parameter matrices $W_i^Q \in R^{d_{model} \times d_k}$,$W_i^k \in R^{d_{model} \times d_k}$, $W_i^V \in R^{d_{model} \times d_v}$ and $W^O \in R^{hd_v \times d_{model}}$.

Multi-Head Attention



$d_{model} = 512$

$W^O \in \mathbb{R}^{hd_v \times d_{model}}$

$W_i^Q \in \mathbb{R}^{d_{model} \times d_k}$

$W_i^k \in \mathbb{R}^{d_{model} \times d_k}$

$W_i^V \in \mathbb{R}^{d_{model} \times d_v}$

Projection

$d_{model} = 512$

$V \in \mathbb{R}^{d_{model}}$

$K \in \mathbb{R}^{d_{model}}$

$Q \in \mathbb{R}^{d_{model}}$

$$MultiHead(Q, K, V) = concat(head_1, \cdots, head_h)W^O$$

$$head_i = Attention(QW_i^Q, KW_i^K, VW_i^V)$$

ベクトルの次元を確認する

$$V \in \mathbb{R}^{d_{model}}$$
$$K \in \mathbb{R}^{d_{model}}$$
$$Q \in \mathbb{R}^{d_{model}}$$

$$W_i^Q \in \mathbb{R}^{d_{model} \times d_k}$$
$$W_i^k \in \mathbb{R}^{d_{model} \times d_k}$$
$$W_i^V \in \mathbb{R}^{d_{model} \times d_v}$$

QW_i^Q：d_{model}次元のベクトルQと$d_{model} \times d_k$ の行列W_i^Qの積
→ d_k次元のベクトル

KW_i^K：d_{model}次元のベクトルKと$d_{model} \times d_k$ の行列W_i^kの積
→ d_k次元のベクトル

VW_i^V：d_{model}次元のベクトルVと$d_{model} \times d_v$ の行列W_i^Vの積
→ d_v次元のベクトル

$$head_i = Attention(QW_i^Q, KW_i^K, VW_i^V)$$

BERT

言語表現モデル

BERTは、2019年に Google が発表した「言語表現モデル」です。

ここでは、「言語表現モデル」という言い方をしていることに注意したらいいと思います。それは、直接には、Google 機械翻訳のように自然言語処理の何らかの具体的な処理を担うモデルではありません。それ自体は、言語のコンピュータ上の「表現」の構成にもっぱら関わるモデルなのです。

“BERT”の意味

BERTは、Bidirectional Encoder Representations from Transformers のイニシャルをとったものです。

そのアーキテクチャーは、前回見たTransformerの前段部のEncoder だけを独立させたようなものです。内部に、Multi headのSelf-Attention Mechanismを抱えています。

そのAttention Mechanismから「表現」を「構成」することが、BERTの「言語表現モデル」としての第一の仕事になります。

Pre-training とFine-tuning

Transformerから出力を担当するDecoderを切り取って、もっぱらEncoder内に「言語表現」を溜め込んで、どうするのでしょう？

BERTでは、膨大な計算時間をかけて、Encoder内部に「言語表現モデル」を構成するまでの、この段階をPre-training と言います。

BERTで外部に対して何かの仕事をしようとするときには、Pre-training で構成された「言語表現」をそのまま使って、具体的なタスクを実行する出力層を一枚かぶせます。これを Fine-tuning と言います。

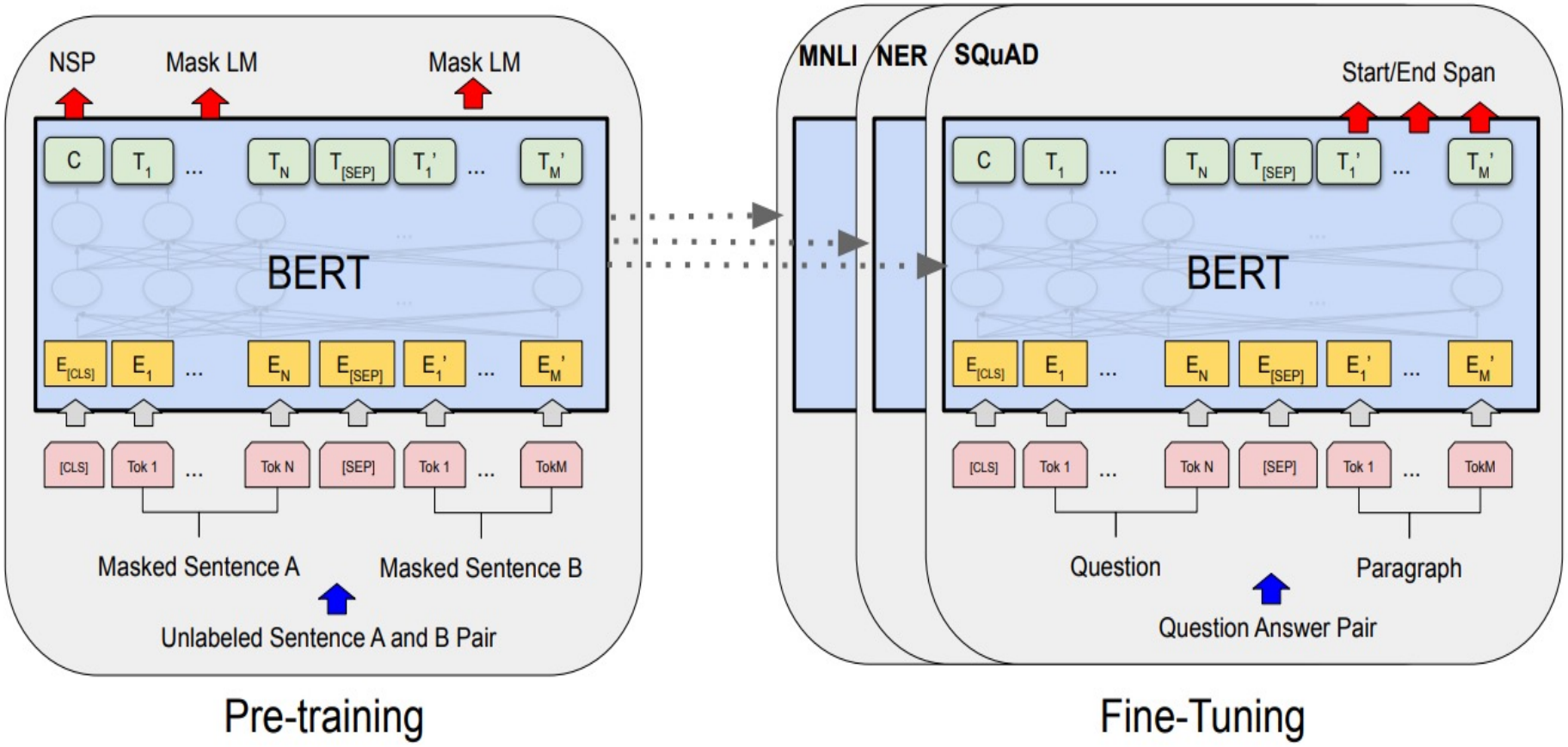
働くBERT

「働くBERT」は、Pre-training + Fine-tuning の形をしています。

ただ、タスクが変わるたびに、長い計算が必要なPre-trainingの段階を繰り返し実行する必要はありません。

「事前に訓練済み」のPre-trained された「言語表現」は、何度も何度も、タスクが変わっても「使い回す」ことができるからです。

Pre-TrainingとFine-Tuning



BERTの「双方向性」

BERTの特徴である Bidirectional 「双方向性」を簡単に説明しておきましょう。

言語の分散表現の基本は、まず、「語」に「ベクトル」を対応させることです。

この計算を実行した、Mikolov-Jeff Deanの「Word2Vec」では、ある語の周辺（前後に関わりなく）にどのような語が出現するかで、語のベクトル表現を求めていきます。(CBOW, N-GRAM)

BERTの「双方向性」

もう一つのアプローチは、ある語の「後ろ」にどのような語が出現するかに注目して、語のベクトル表現を求めることです。これをleft-to-right と呼ぶことにしましょう。

当然、ある語の「前」に、どのような語が出現するかに注目して、語のベクトル表現を見つけるやり方もあり得ます。これを、right-to-left と呼ぶことにしましょう。

BERTは、このleft-to-right とright-to-leftの双方向で語のベクトル表現を見つけます。そのために、文の中の一つの語をランダムに選んで、マスクをかけてその語を推定させる訓練を繰り返します。双方向から語の推定をしていくのは有効です。

文と文の関係

BERTにはもうひとつ際立った特徴があります。
それは、二つの文の関係を把握しようとしていることです。

具体的には、ある文Aともう一つの文Bが与えられたとき、文Aのうしろには文Bがつながるということを学習させます。

こうした文と文の関係は、語の意味の表現(Word2Vex)や文の意味の表現(Sentence2Sentence)のような、個別の語、個別の文の意味表現のレベルでは、把握することはできません。

ただ、実際の言語理解では、こうした文の繋がりが、重要な役割を果たしています。

文と文の関係

例えば、

Sentence A = The man went to the store.

Sentence B = He bought a gallon of milk.

だとすると、この二つの文は、つながっていると判断できますので、"IsNext"というラベルを出力するように訓練します。

Sentence A = The man went to the store.

Sentence B = Penguins are flightless.

だとすると、この二つの文は関連が薄いので、"NotNext" というラベルを出力するように訓練します。

BERTがPre-trainingで行うこと

BERTのPre-trainingでは、次の二つのことを、徹底して学習します。

- 双方向での語の意味の表現
- 二つの文が与えられたとき、それが関連しているかの判断

BERT論文 タイトル

BERT論文のタイトルは、"BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding" というのですが、そこでの "Language Understanding" というのは、単なる「翻訳モデル」を超えた「言語理解」を目指すという方向を示唆しています。

二つの文の関係を視野に入れたことで、BERTの能力は大きく拡張可能なものになりました。

Fine-tuning Taskの例

もっとも、そうした具体的な課題での「言語理解」のタスクは、Fine-tuningの層で実行されることになります。スライドでは、いくつかのFine-tuning層で実行されるタスクの例を挙げておきました。

- **GLUE**: The General Language Understanding Evaluation
- **SQuAD v1.1**: The Stanford Question Answering Dataset
- **SWAG**: Situations With Adversarial Generations
- **MNLI**: Multi-Genre Natural Language Inference
- **QNL**I: Question Natural Language Inference

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

Jacob Devlin, Ming-Wei Chang, Kenton Lee, Kristina Toutanova

https://arxiv.org/abs/1810.04805

2019年

Abstract

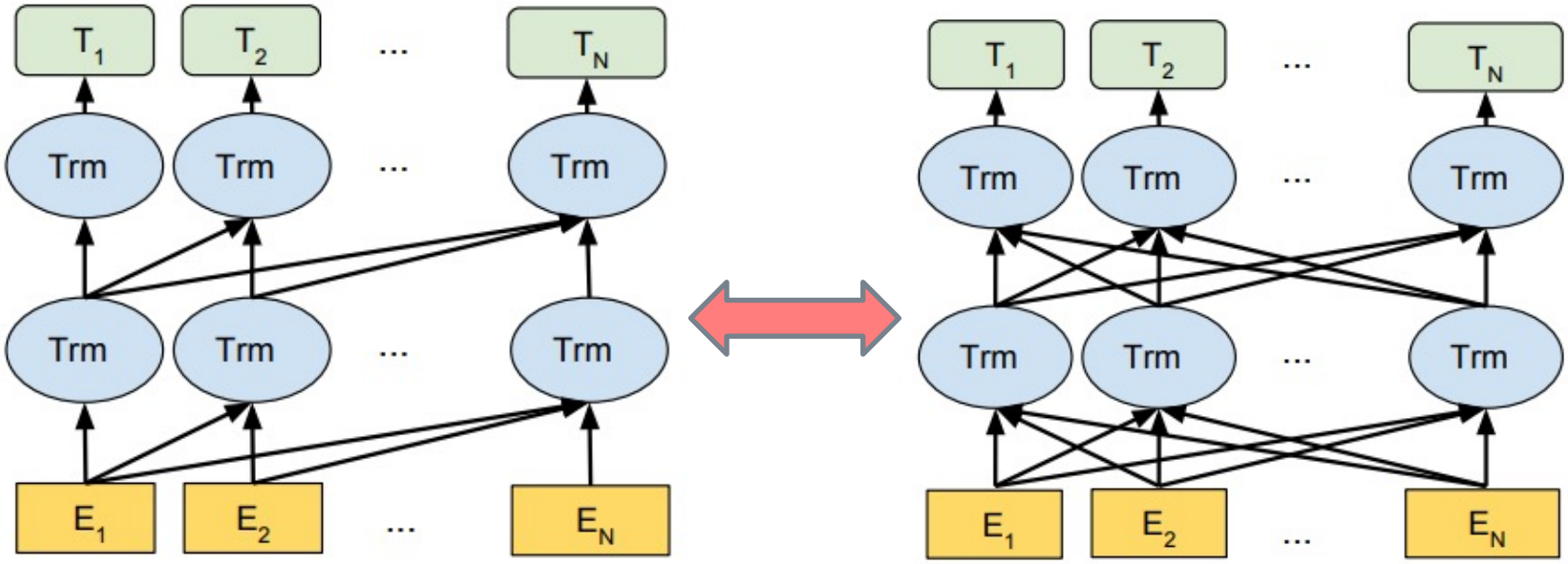
我々は、BERT（Bidirectional Encoder Representations from Transformersの略）と呼ばれる新しい言語表現モデルを紹介する。

最近の言語表現モデルとは異なり、BERTは、全層で左右両方の文脈を共同で条件付けることにより、ラベルのないテキストから深い双方向表現を事前に学習するように設計されている。

その結果、事前に訓練されたBERTモデルは、たった1つの出力層を追加するだけでファイ・チューニングが可能となり、タスク固有のアーキテクチャを大幅に変更することなく、質問応答や言語推論などの幅広いタスクに対して最先端のモデルを作成することができる。

Pre-training モデルの違い

BERT uses a bidirectional Transformer. OpenAI GPT uses a left-to-right Transformer. BERT representations are jointly conditioned on both left and right context in all layers.



OpenAI GPT ： left-to-right

BERT: bidirectional

BERTでの意味の分散表現

In this work, we denote the number of layers (i.e., Transformer blocks) as L, the hidden size as H, and the number of self-attention heads as A.[3] We primarily report results on two model sizes: **BERT$_{\text{BASE}}$** (L=12, H=768, A=12, Total Parameters=110M) and **BERT$_{\text{LARGE}}$** (L=24, H=1024, A=16, Total Parameters=340M).

we denote input embedding as E, the final hidden vector of the special `[CLS]` token as $C \in \mathbb{R}^H$, and the final hidden vector for the i^{th} input token as $T_i \in \mathbb{R}^H$.

Pre-traning BERTのタスク
Masked LM

Masked LM

- 80% of the time: Replace the word with the [MASK] token,
  e.g., my dog is hairy → my dog is [MASK]
- 10% of the time: Replace the word with a random word,
  e.g., my dog is hairy → my dog is apple
- 10% of the time: Keep the word unchanged,
  e.g., my dog is hairy → my dog is hairy.

The purpose of this is to bias the representation towards the actual observed word.

Pre-traning BERTのタスク
NSP: Next Sentence Prediction

NSP: Next Sentence Prediction

- QA: Question Answering
- NLI: Natural Language Inference

Many important downstream tasks such as Question Answering (QA) and Natural Language Inference (NLI) are based on understanding the relationship between two sentences, which is not directly captured by language modeling.

Next Sentence Prediction

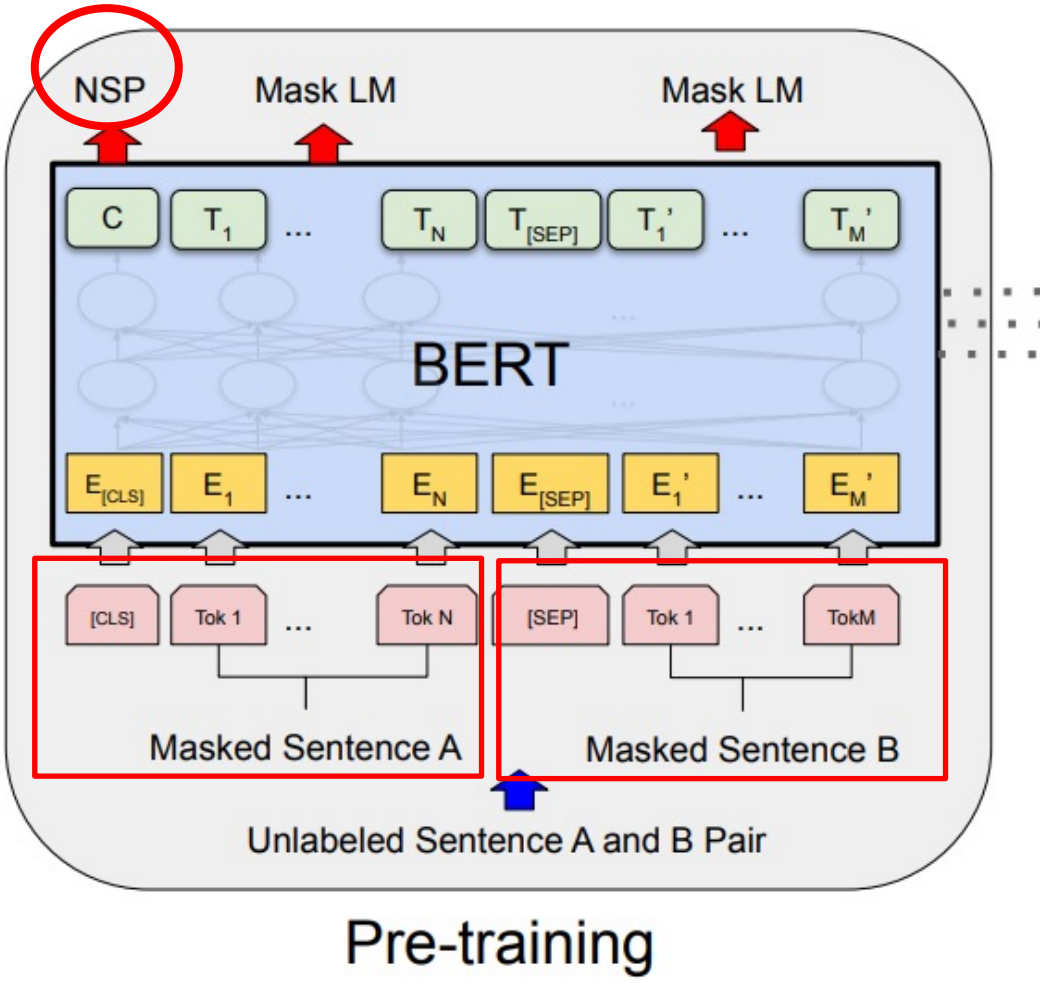
Input = [CLS] the man went to [MASK] store [SEP]
he bought a gallon [MASK] milk [SEP]

Label = IsNext

Input = [CLS] the man [MASK] to the store [SEP]
penguin [MASK] are flight ##less birds [SEP]

Label = NotNext

Pre-traning BERTのタスク
NSPの入力:Sentence Pair



- NSP:
  Next
  Sentence
  Prediction

IsNext/NotNext

Fine-tuning BERTのタスク
GLUE

GLUE: The General Language Understanding Evaluation

a collection of diverse natural language understanding tasks.

To fine-tune on GLUE, we represent the input sequence (for single sentence or sentence pairs) as described in Section 3, and use the final hidden vector $C \in R^H$ corresponding to the first input token ([CLS]) as the aggregate representation.

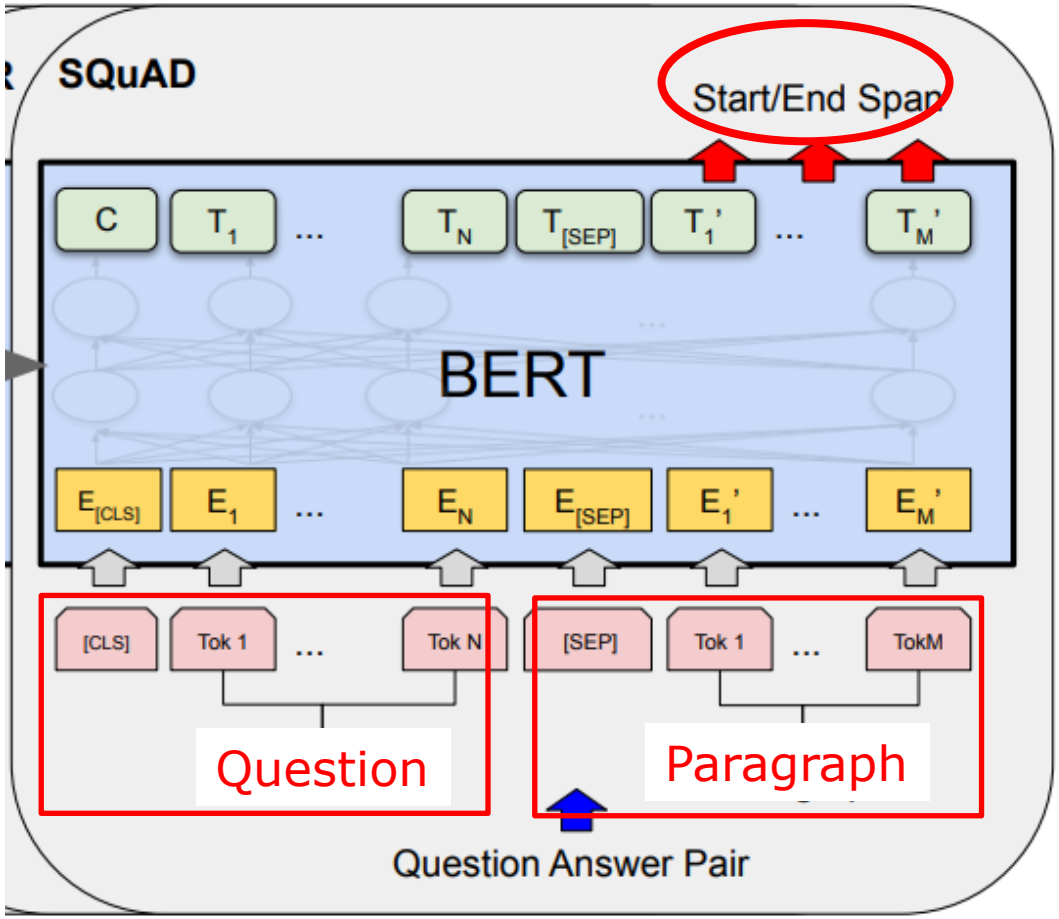
Fine-tuning BERTのタスク
SQuAD v1.1

SQuAD v1.1: The Stanford Question Answering Dataset

a collection of 100k crowd sourced question/answer pairs.

Given a question and a passage from Wikipedia containing the answer, the task is to predict the answer text span in the passage.

SQuAD



in the question answering task, we represent the input question and passage as a single packed sequence, with the question using the A embedding and the passage using the B embedding. We only introduce a start vector $S \in R^H$ and an end vector $E \in R^H$ during fine-tuning.

SQuAD v2.0

The SQuAD 2.0 task extends the SQuAD 1.1 problem definition by allowing for the possibility that no short answer exists in the provided paragraph, making the problem more realistic.

We use a simple approach to extend the SQuADv1.1 BERT model for this task. We treat questions that do not have an answer as having an answer span with start and end at the [CLS] token. The probability space for the start and end answer span positions is extended to include the position of the [CLS] token.

SWAG

SWAG: Situations With Adversarial Generations, contains 113k sentence-pair completion examples that evaluate grounded commonsense inference.

When fine-tuning on the SWAG dataset, we construct four input sequences, each containing the concatenation of the given sentence (sentence A) and a possible continuation (sentence B).

The only task-specific parameters introduced is a vector whose dot product with the [CLS] token representation C denotes a score for each choice which is normalized with a softmax layer

GLUE Benchmark Experient

MNLI Multi-Genre Natural Language Inference

is a large-scale, crowd sourced entailment classification task (Williams et al., 2018). Given a pair of sentences, the goal is to predict whether the second sentence is an entailment, contradiction, or neutral with respect to the first one.

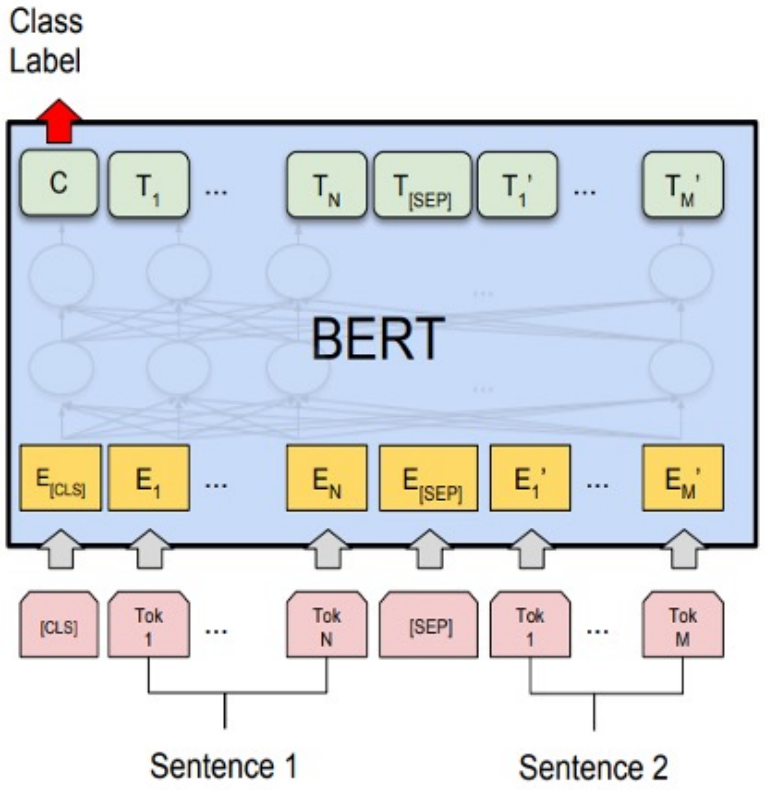
QQP Quora Question Pairs

is a binary classification task where the goal is to determine if two questions asked on Quora are semantically equivalent (Chen et al., 2018).

QNLI Question Natural Language Inference

Is a version of the Stanford Question Answering Dataset (Rajpurkar et al., 2016) which has been converted to a binary classification task (Wang et al., 2018a). The positive examples are (question, sentence) pairs which do contain the correct answer, and the negative examples are (question, sentence) from the same paragraph which do not contain the answer.

Multi-Genre Natural Language Inference MNL1



(a) Sentence Pair Classification Tasks: MNLI, QQP, QNLI, STS-B, MRPC, RTE, SWAG

Given a pair of sentences, the goal is to predict whether the second sentence is an entailment, contradiction, or neutral with respect to the first one.

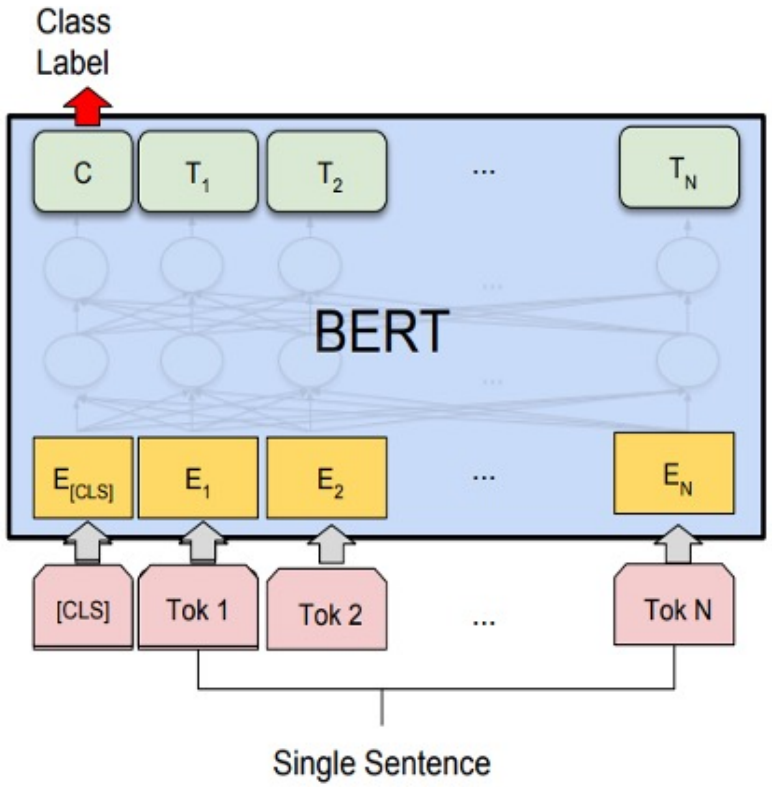
SST-2 The Stanford Sentiment Treebank

is a binary single-sentence classification task consisting of sentences extracted from movie reviews with human annotations of their sentiment (Socher et al., 2013).

CoLA The Corpus of Linguistic Acceptability

Is a binary single-sentence classification task, where the goal is to predict whether an English sentence is linguistically “acceptable” or not (Warstadt et al., 2018).

The Stanford Sentiment Treebank SST-2



(b) Single Sentence Classification Tasks: SST-2, CoLA

binary single-sentence classification task consisting of sentences extracted from movie reviews with human annotations of their sentiment

STS-B The Semantic Textual Similarity Benchmark

is a collection of sentence pairs drawn from news headlines and other sources (Cer et al.,2017).

They were annotated with a score from 1 to 5 denoting how similar the two sentences are in terms of semantic meaning

MRPC Microsoft Research Paraphrase Corpus

consists of sentence pairs automatically extracted from online news sources, with human annotations for whether the sentences in the pair are semantically equivalent (Dolan and Brockett, 2005).

CoNLL-2003 Named Entity Recognition

results. Hyperparameters were selected using the Dev set. The reported Dev and Test scores are averaged over 5 random restarts using those hyperparameters.