現場で使えるディープラーニング基礎講座

DAY7

SkillUP AI

前回の復習

- 前回は何を学びましたか？

自然言語処理における深層学習

目次

自然言語処理と深層学習

自然言語処理における深層学習

- 人間がコミュニケーションに用いている言葉を対象とする処理を自然言語処理と呼ぶ。
  - 自然言語以外の言語の例
    - コンピュータ言語などの形式言語
    - プログラミング言語
    - エスペラントのような人工言語
- 自然言語処理の分野でも深層学習を用いた手法が多数提案されている。

伝統的な自然言語処理

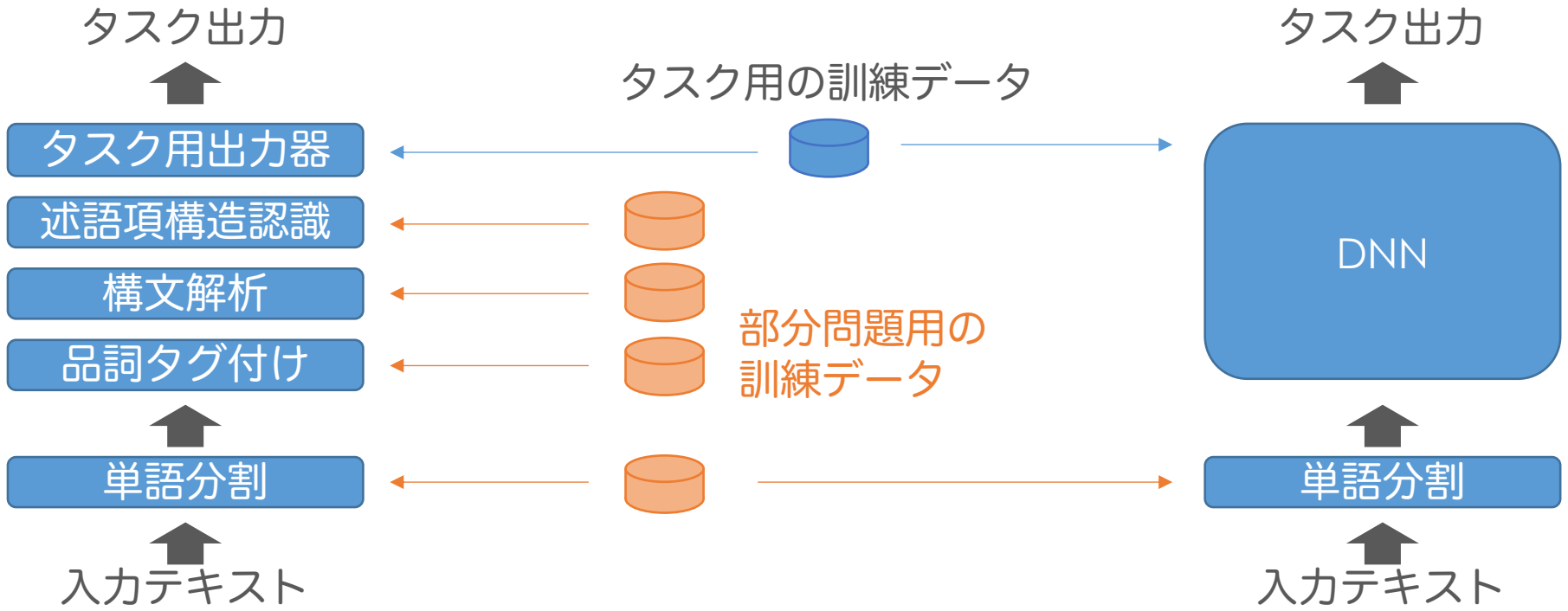
- 自然言語処理で典型的なタスクには、文章分類、機械翻訳、文章要約、質問応答、対話などがある。
- 伝統的な自然言語処理では、これらのタスクを部分問題に分解していた。
- 主な部分問題を以下に示す。

部分問題	処理内容
品詞タグ付け	単語に名詞・動詞などの文法的な役割分類(品詞)を付与する処理
単語分割	日本語などの単語に分けられていないテキストを単語列に分割する処理
語義曖昧性解消	複数の語義をもつ単語の語義を特定する処理
固有表現抽出	人名・地名・日付などを抽出する処理
構文解析	文法に基づく文の木構造を構築する処理
述語項構造認識	述語を中心とした意味構造を抽出する処理

参考:深層学習による自然言語処理, 坪井ら, 講談社

深層学習による自然言語処理システム

- 従来の自然言語処理システムでは、部分問題用学習器を組み合わせることにより、タスクを解いていた。
- 深層学習を用いた自然言語処理システムでは、1つの学習器でタスクを解くようにモデルが構築される。



従来の自然言語処理システムの例

深層学習による自然言語処理システムの例

深層学習による自然言語処理システム

- 深層学習を用いた自然言語処理システムでは、再帰型ニューラルネットワーク(RNN)がよく用いられる。
- 一方で、RNNを用いずに機械翻訳問題を解くというモデルも提案されている。
  - Ashish Vaswani et al. Attention is All You Need, NIPS 2017. https://arxiv.org/pdf/1706.03762.pdf
- 単語の埋め込み行列を求める際には、全結合型ニューラルネットワークが用いられることが多い。
  - 例、word2vec

自然言語処理に関わる代表的なできごと

- 東西冷戦下では、英語-ロシア語の機械翻訳が注目される。
- 1964年~1966年、心理カウンセラーのように振る舞う対話型プログラムELIZAがジョセフ・ワイゼンバウムによって開発される。
- 2011年、IBMが開発した人工知能ワトソンが、アメリカのクイズ番組「ジョパディー」に出演し、歴代の人間チャンピオンと対戦し勝利。
- 東大入試合格を目指す人工知能東ロボくん。国立情報学研究所が中心となって、2011年~2016年まで開発が続けられ、偏差値57.8をマークし、私立大学に合格できるレベルまで達した。
- 2016年、Googleのニューラル機械翻訳がサービス開始。
- 2010年代、スマートスピーカーが普及(Apple Siri:2011年、Amazon Alexa:2014年、Google Home:2016年)

自然言語処理の基礎

形態素解析

- 自然言語を形態素まで分解する技術。
- 形態素とは、何らかの意味をもつ最小限の文字の集まり。
- 日本語での検索エンジンや自然言語処理で極めて重要な技術。
- MeCabやJUMAN、ChaSenなどが代表的な形態素解析エンジン。

すももももももものうち

すもも	も	もも	も	もも	の	うち
名詞、一般	助詞、係助詞	名詞、一般	助詞、係助詞	名詞、一般	助詞、連体化	名詞、非自立、副詞可能

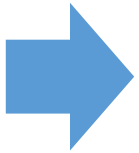
n-gram

- n-gramとは、 隣り合って出現したn個の単語のこと。文字n-gramもあるが、ここでは単語n-gramのことを指している。
- n=1の場合は、1-gram(ユニグラム)。
  - 吾輩, は, 猫, で, ある
- n=2の場合は、2-gram(バイグラム) 。
  - 吾輩は, は猫, 猫で, である
- n=3の場合は、3-gram(トリグラム) 。
  - 吾輩は猫, は猫で, 猫である

バッグオブワーズ

- バッグオブワーズ(bag-of-words)とは、文章における単語の出現頻度を数えて、頻度ベクトルで表現する方法。
- 単語(words)をバラバラにして袋(bag)に入れた状態に例えられている。
- 頻度ベクトルには、語順の情報は残らない。
- 実際には、出現頻度が極端に低い単語や一般的すぎる単語を外してベクトル化する。
- n-gramで数えることもある。

すももももももものうち
すいかもももまるいくだもの



左から、
{すもも、もも、すいか、も、の、まるい、うち、くだもの}
の出現頻度

[1, 2, 0, 2, 1, 0, 1, 0]
[0, 1, 1, 2, 0, 1, 0, 1]

頻度ベクトル

分布仮説

- 分布仮説(distributional hypothesis)とは、「単語の意味は、その単語が出現した際の周囲の単語によって決まる」という考え方。
- 1950年代に提案された。
- 後述するword2vecは、この分布仮説を用いて分散表現を獲得していると解釈できる。

私　は　今日　会社　へ　行く

会社 という単語の意味は、周辺の単語(文脈) から決まる

[演習] MeCabによる形態素解析

- 7_1_morphological _analysis.ipynb
  - MeCabはオープンソースの高速な形態素解析エンジンです。
  - MeCabを用いて形態素解析を行なってみましょう。

word2vec

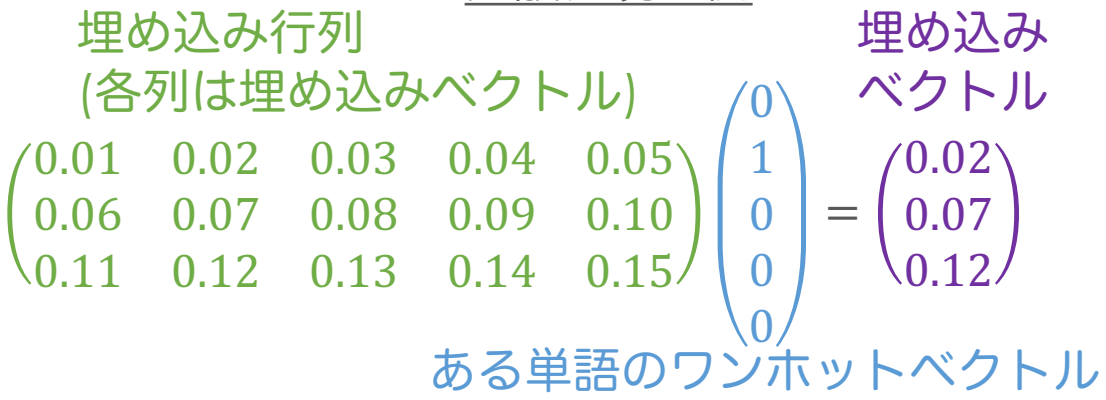
単語を数学的に表現するには？

- 自然言語を計算機で扱うには、記号である「単語」を数学的に表現する必要がある。
- 単語を数学的に表現するにはどうすればいいか？
- 例えば、ワンホットベクトルで表現するという方法が考えられる。
  - この方法の場合、ベクトル同士の演算を行うことに意味がない。
  - しかも、語彙数が増えると非常に高次元なベクトルになってしまう。
- もっと賢い方法はないか？

分散表現

- 分散表現(単語を強調する場合は単語分散表現と呼ばれる)とは、単語を計算機上で扱うための方法の1つ。単語埋め込み(word embeddings)とも呼ばれる。
- 単語間の関連性や類似度を考慮したベクトル(埋め込みベクトル)で、ある単語を表現する。
- 埋め込みベクトルどうしは、足し算や引き算が可能。
- 単語が5つだけの場合の分散表現の例を以下に示す。
- 単語埋め込み行列を求めるには、word2vecなどを用いる。

分散表現の例



単語のワンホットベクトルは、実際の問題では数万次元になったりする。一方、埋め込みベクトルは数百次元くらいに設定する。ある単語の埋め込みベクトルには、ある単語の情報が埋め込まれていると解釈できる。

分散表現

- 埋め込みベクトルは、単語間の類似度を計算する目的でも使われるが、次の何らかの分析に利用されることも多い。
- 例えば、回帰モデルやクラス分類モデルの入力に埋め込みベクトルを用いるという活用が考えられる。

word2vec

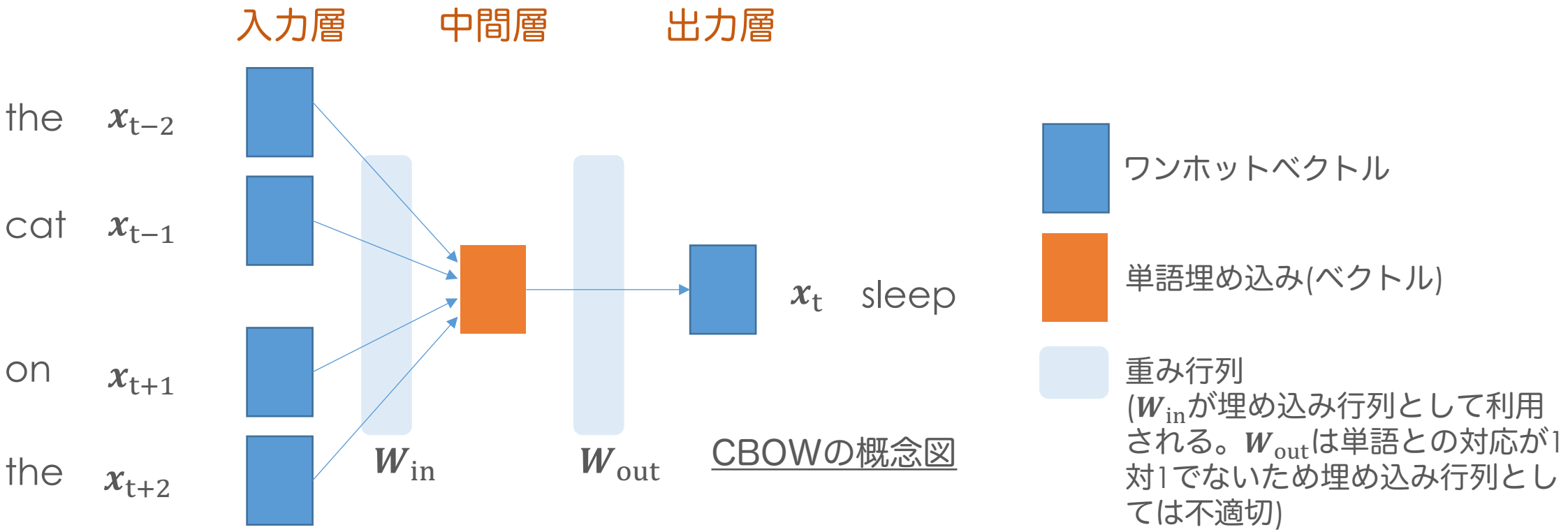
- 埋め込み行列を学習することができるツール。
- word2vecの公式ウェブサイト
  - https://code.google.com/archive/p/word2vec/
- word2vecをPythonから利用するには、gensimなどのライブラリを使用するのがよい。
- 「word2vecで求めた埋め込み行列を用いると、王 – 男 + 女 = 女王 のような演算が可能になる」と紹介されることが多い。
- 原著論文
  - Tomas Mikolov et al. Distributed Representations of Words and Phrases and their Compositionality. https://arxiv.org/pdf/1310.4546.pdf
  - Tomas Mikolov et al. Efficient Estimation of Word Representations in Vector Space. https://arxiv.org/pdf/1301.3781.pdf

word2vec

- word2vecには、埋め込み行列を学習する方法として、Continuous Bag-of-Words (CBOW) と skip-gramの2つのニューラルネットワークが用意されている。
- CBOWは、前後の単語から、ある単語を予測するためのニューラルネットワーク。
  - 例えば、”the cat sleep on the mat”という文章がある場合、 “cat” と “on” から”sleep” を予測できるように学習を行う。
- skip-gram とは、ある単語が与えられた時に、その周辺の単語を予測するためのニューラルネットワーク。
  - 例えば、”the cat sleep on the mat”という文章がある場合、 “sleep”から”cat”と”on”を予測できるように学習を行う。
- CBOWもskip-gram も、分布仮説を用いて分散表現を獲得していると解釈できる。

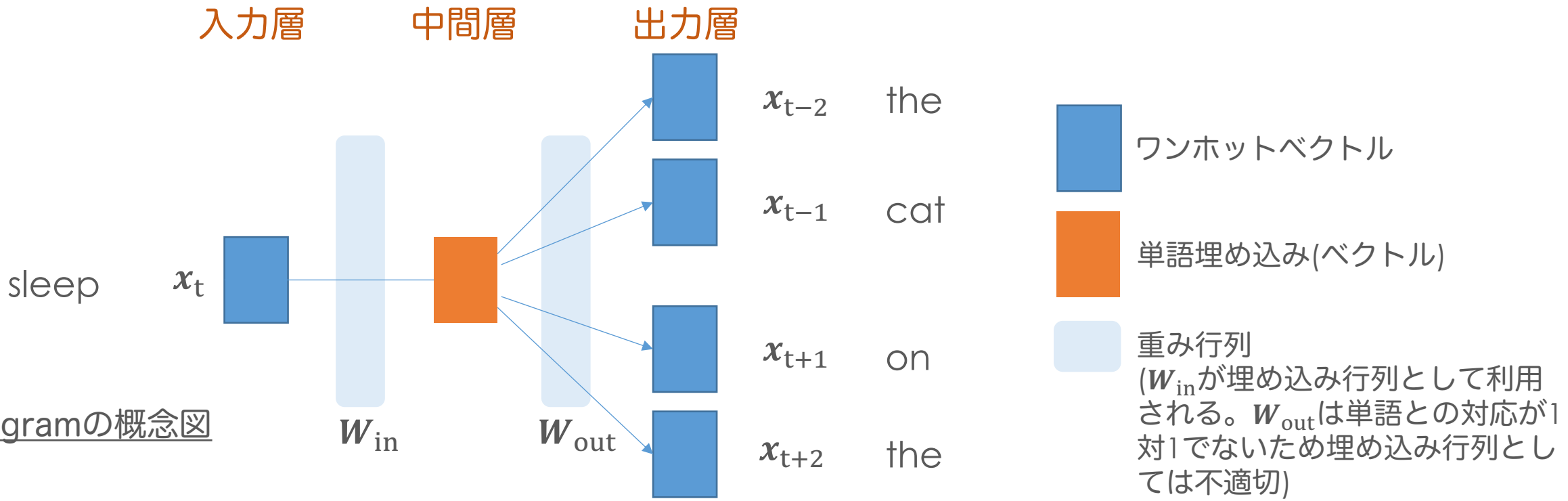
Continuous Bag-of-Words

- word2vecのアルゴリズムの1つ。
- 前後の単語から対象の単語を予測するニューラルネットワーク。
- 学習にかかる時間がskip-gramよりも短い。



skip-gram

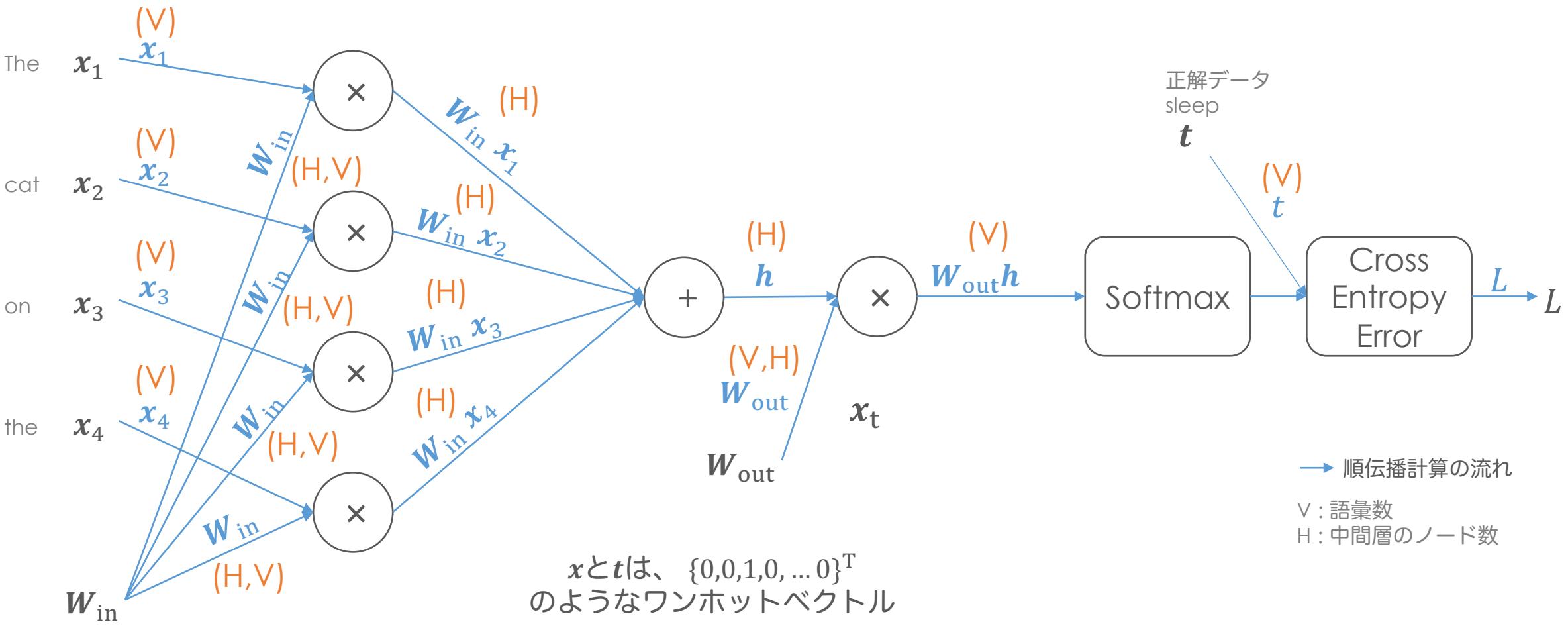
- word2vecのアルゴリズムの1つ。
- ある単語から、前後の単語を予測するニューラルネットワーク。
- CBOWに比べ、学習に時間がかかるが、精度が高くなる傾向がある。



skip-gramの概念図

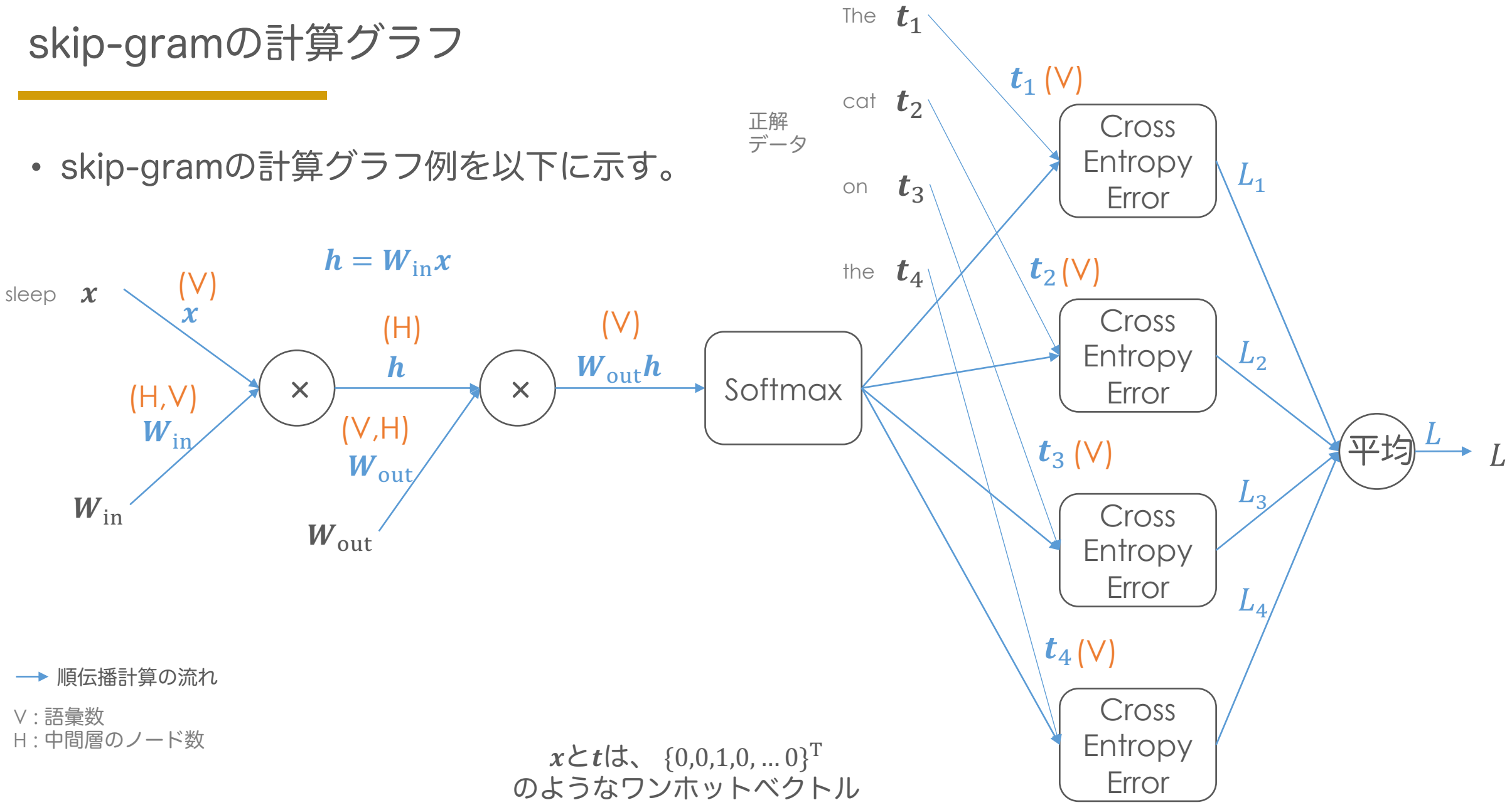
Continuous Bag-of-Wordsの計算グラフ

- CBOWの計算グラフ例を以下に示す。



skip-gramの計算グラフ

- skip-gramの計算グラフ例を以下に示す。



コサイン類似度

- コサイン類似度とは、ベクトルどうしの類似度を測る指標。
- 埋め込みベクトルどうしの類似度を測る際にも使用される。

<コサイン類似度>

$$\text{similarity}(\boldsymbol{x}, \boldsymbol{y}) = \frac{\boldsymbol{x} \cdot \boldsymbol{y}}{\|\boldsymbol{x}\|\|\boldsymbol{y}\|} = \frac{x_1 y_1 + x_1 y_1 \cdots x_n y_n}{\sqrt{x_1^2 + x_2^2 \cdots x_n^2}\sqrt{y_1^2 + y_2^2 \cdots y_n^2}} = \cos\theta$$

$\boldsymbol{x} = \{x_1, x_2, \cdots, x_n\}$
$\boldsymbol{y} = \{y_1, y_2, \cdots, y_n\}$

[演習] word2vecを使って単語の分散表現を獲得する

- 7_2_word2vec.ipynb
  - word2vecを使って単語の分散表現を求めましょう。

Any Questions?

系列変換モデル

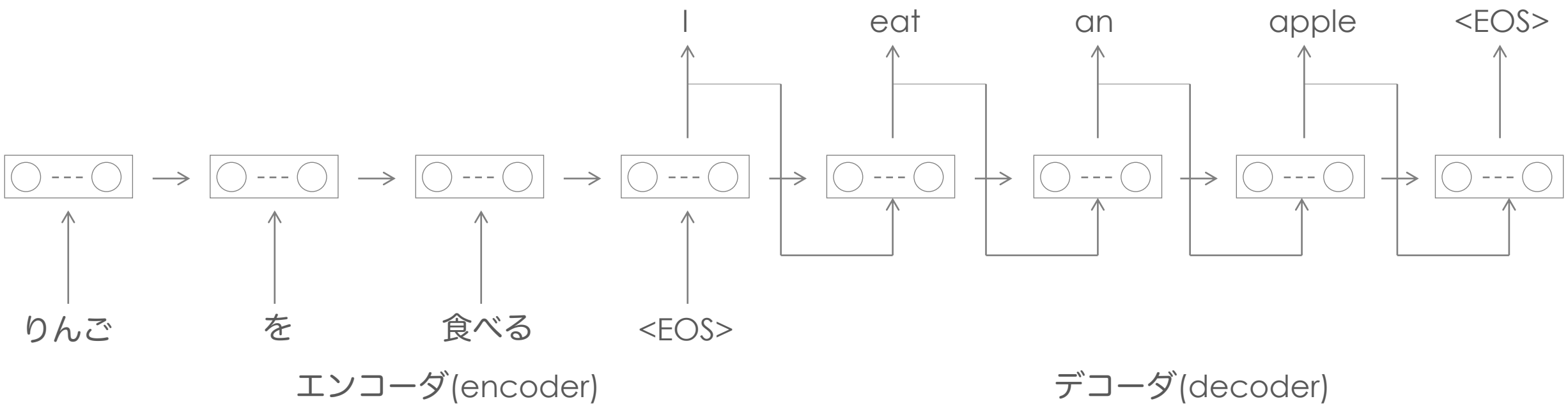
系列変換モデル

- 単語列などの系列(sequence)を受け取り、別の系列へ変換するモデルのことを系列変換モデル(sequence-to-sequence model, seq2seqモデル)という。
- 主に自然言語処理で用いられる。
- RNNエンコーダ・デコーダとも呼ばれる。
- 適応されるタスクの例
  - 機械翻訳 (翻訳元の言語から翻訳先の言語への変換)
  - 対話 (相手の発言から自分の発言への変換)
  - 質問応答 (質問文から返答文への変換)
  - 文書要約 (元文書から要約文への変換)

系列変換モデル

- 入力系列の長さと出力系列の長さは、一致していなくてもよい。
- 系列の長さは、データによって異なっていてもよい。

系列変換モデルの概念図



系列変換モデルの呼ばれ方

- 系列変換モデルは、seq2seqモデルと呼ばれる場合があったり、RNNエンコーダ・デコーダと呼ばれる場合があったりする。
- これは、両者の起源とする論文が異なるためである。
- 両者の概念はほぼ同じであるため、実務上は区別なく呼ばれることが多い。
- seq2seqモデル
  - Ilya Sutskever, Oriol Vinyals, Quoc V. Le. Sequence to Sequence Learning with Neural Networks.2014. https://papers.nips.cc/paper/5346-sequence-to-sequence-learning-with-neural-networks.pdf
- RNNエンコーダ・デコータ
  - Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, Yoshua Bengio. Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation. 2014. https://www.aclweb.org/anthology/D14-1179

[演習] seq2seqモデルの実装

- 7_3_seq2seq.ipynb
  - seq2seqモデルを実装しましょう。
  - RNN部分には、LSTMを用います。

アテンション

アテンション

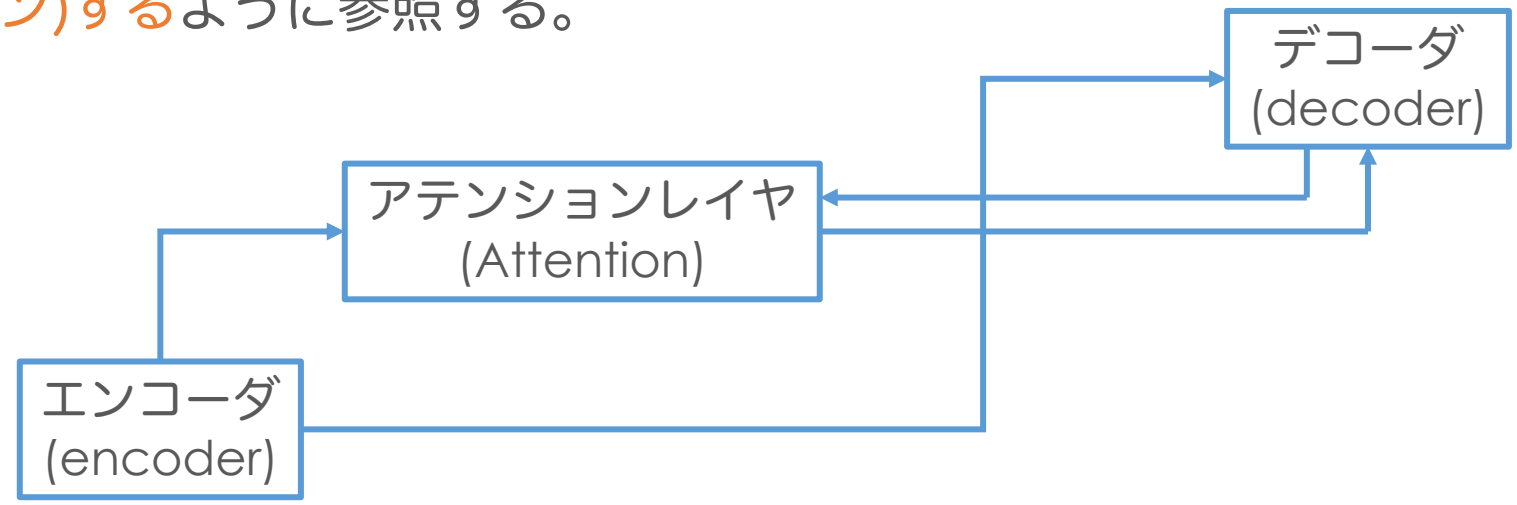
- 系列変換モデルでは、系列情報をRNNで固定長のベクトルに変換するが、系列が長くなると、LSTMのような仕組みを使ってもうまく学習させるのは難しい。
  - 系列が長くなると、最初に入力された情報がデコーダ側まで伝播しづらくなる。
- この課題に対処する方法として、アテンションという仕組みがある。
- アテンションの仕組みを最初に提案した論文
  - Dzmitry Bahdanau, Kyunghyun Cho, Yoshua Bengio. Neural Machine Translation by Jointly Learning to Align and Translate. https://arxiv.org/pdf/1409.0473.pdf

ソフト・アテンションとハード・アテンション

- ソフト・アテンション
  - ソフトマックス関数などで確率分布を求め、その確率分布を用いた重み付き平均をとることによって、データに注目する方法。
  - seq2seqモデルで用いるアテンションは、ソフト・アテンション。
- ハード・アテンション
  - ソフトマックス関数などで確率分布を求め、その確率分布に従って無作為抽出された1点だけに注目する方法。
  - 無作為抽出が入っていると微分不可能なので、何らかの方法を用いて勾配を推定する必要がある。

系列変換モデルにおけるアテンション

- 系列変換モデルにおいて、通常、アテンションは、一つのレイヤとして扱われる。
- アテンションレイヤは、エンコーダとデコーダの間に位置付けられる。
- 通常の系列変換モデルでは、エンコーダの最終中間状態だけがデコーダに伝わるが、アテンション付き系列変換モデルでは、全時刻の中間状態を参照する。
  - その際、全時刻の中間状態を等価に参照するのではなく、重要な中間状態をより重視(アテンション)するように参照する。



系列変換モデルにおけるアテンションの仕組み

- 系列変換モデルにおけるアテンションの仕組みを、以下の論文で提案されたグローバル・アテンションモデルを例に説明する。
- 参照論文
  - Minh-Thang Luong, Hieu Pham, Christopher D. Manning. Effective Approaches to Attention-based Neural Machine Translation. https://arxiv.org/pdf/1508.04025.pdf
- 同様の解説が以下の文献にも記載されているため、そちらも参照されたい。
  - 坪井裕太,海野裕也,鈴木潤. 深層学習による自然言語処理. 講談社. 4章

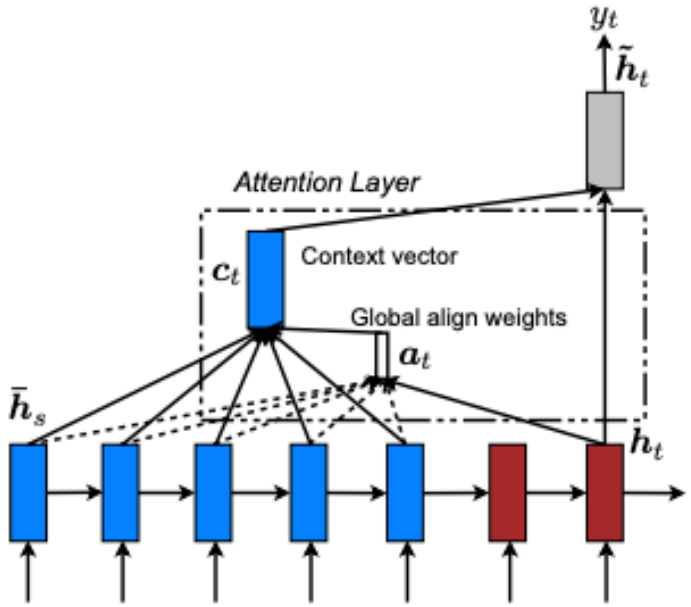


Figure 2: **Global attentional model** – at each time step t, the model infers a *variable-length* alignment weight vector a_t based on the current target state h_t and all source states $\bar{h}_s$. A global context vector c_t is then computed as the weighted average, according to a_t, over all the source states.

参照論文より引用

系列変換モデルにおけるアテンションの仕組み

翻訳タスクであればあれば、例えば $\boldsymbol{x}$ が日本語、 $\mathbf{y}$ が英語

- 入力系列を$\boldsymbol{x} = \{x_1, \cdots, x_I\}$、目標系列を $\mathbf{y} = \{y_1, \cdots, y_J\}$とする。
- エンコーダが計算する各時刻の中間状態ベクトルを$\{\boldsymbol{h}_1^{(s)}, \cdots, \boldsymbol{h}_i^{(s)}, \cdots, \boldsymbol{h}_I^{(s)}\}$とおく。

sはsource、tはtargetを意味する

- デコーダが計算する各時刻の中間状態ベクトルを$\{\boldsymbol{h}_1^{(t)}, \cdots, \boldsymbol{h}_j^{(t)}, \cdots, \boldsymbol{h}_J^{(t)}\}$とおく。
- 系列変換モデルでは、最終的に、 $\boldsymbol{h}_j^{(t)}$を用いて、 y_jの確率分布を求めたい。

$$p\left(y_j | \boldsymbol{y}_{<j}, \boldsymbol{x}\right) = \mathrm{softmax}\left(\boldsymbol{W}^{(t)} \boldsymbol{h}_j^{(t)}\right)$$

$\boldsymbol{W}^{(t)}$:パラメータ

$\mathrm{softmax}$：ソフトマックス関数

系列変換モデルにおけるアテンションの仕組み

- エンコーダ側のRNNの遷移関数を$\psi^{(s)}$とすると、エンコーダの中間状態ベクトルは以下のように再帰的に計算される。

$$\boldsymbol{h}_i^{(s)} = \psi^{(s)}\left(\boldsymbol{x}_i, \boldsymbol{h}_{i-1}^{(s)}\right)$$

$\psi^{(s)}$は、LSTMレイヤなど

- 通常の系列変換モデルでは、最後の中間状態ベクトル$\boldsymbol{h}_I^{(s)}$のみを使って、デコーダの計算を行う
- この時、例えば、 $\boldsymbol{x}_1$の情報は、 $\psi^{(s)}$がI回適応される間、ずっと各$\boldsymbol{h}_i^{(s)}$で保持される必要がある。
- もっと直接的に入力系列の情報をデコーダ側に伝える方法はないだろうか？

  -> これがアテンションの動機

系列変換モデルにおけるアテンションの仕組み

- デコーダの中間状態ベクトル$\boldsymbol{h}_j^{(t)}$もエンコーダと同様に以下で計算される。

$$\boldsymbol{h}_j^{(t)} = \psi^{(t)}\left(\boldsymbol{y}_i, \boldsymbol{h}_{j-1}^{(t)}\right)$$

$\psi^{(t)}$は、LSTMレイヤなど

- この後、デコーダは、 $\boldsymbol{h}_j^{(t)}$を利用して、 目標系列の確率分布$p\left(y_j|\boldsymbol{y}_{<j}, \boldsymbol{x}\right)$を計算するが、ここで、エンコーダが出力した中間状態ベクトル$\boldsymbol{h}_i^{(s)}$を直接利用することを考える。

系列変換モデルにおけるアテンションの仕組み

- デコーダがj番目の単語を予測する時に、エンコーダのi番目の中間状態ベクトル$\boldsymbol{h}_i^{(s)}$の重要度を示すスカラー値a_iを計算し、a_{ij}による$\boldsymbol{h}_i^{(s)}$の重み付き平均$\boldsymbol{c}_j$を計算する。

$$\boldsymbol{c}_j = \sum_i^I a_{ij}\boldsymbol{h}_i^{(s)}$$ 加重平均された中間状態

$$a_{ij} = \frac{\exp(e_{ij})}{\sum_{i=1}^I \exp(e_{ij})}, a_{ij} \in \mathbb{R}$$ 中間状態の重要度

$$e_{ij} = \Omega\left(\boldsymbol{h}_i^{(s)}, \boldsymbol{h}_j^{(t)}\right), e_{ij} \in \mathbb{R}$$

- 例えば、3番目の入力単語$\boldsymbol{x}_3$だけが重要であれば、 a_{3j}が1に近く、それ以外のa_{ij}がほとんど0に近くなる。
- a_{ij}は、対象となる複数のベクトルの中から、重要な情報を選別するような役割を担うことになる。

系列変換モデルにおけるアテンションの仕組み

- 関数Ωについて、参照論文では以下のような複数の関数を検討している。

$$\Omega\left(\boldsymbol{h}_i^{(s)}, \boldsymbol{h}_j^{(t)}\right) = \begin{cases} \boldsymbol{h}_i^{(s)} \cdot \boldsymbol{h}_j^{(t)} \\ \boldsymbol{h}_i^{(s)} \cdot \boldsymbol{W}\boldsymbol{h}_j^{(t)} \\ \boldsymbol{v} \cdot \tanh(\boldsymbol{W}[\boldsymbol{h}_i^{(s)}; \boldsymbol{h}_j^{(t)}]) \end{cases}$$

内積

デコーダ側中間状態ベクトルに重みをかけてから内積

エンコーダ側中間状態ベクトルとデコーダ側中間状態ベクトルを結合してから重みをかけて、 tanhを通し、重みベクトル$\boldsymbol{v}$と内積

$\boldsymbol{W}, \boldsymbol{v}$：パラメータ

[;]：結合(concatenate)の記号

系列変換モデルにおけるアテンションの仕組み

- デコーダがj番目の単語を予測する際、$\boldsymbol{h}_j^{(t)}$の代わりに以下で定義される$\widetilde{\boldsymbol{h}}_j^{(t)}$を利用する。

  $\widetilde{\boldsymbol{h}}_j^{(t)} = \tanh(\boldsymbol{W}_c\ [\boldsymbol{c}_j; \boldsymbol{h}_j^{(t)}])$

  $\boldsymbol{W}_c$：パラメータ

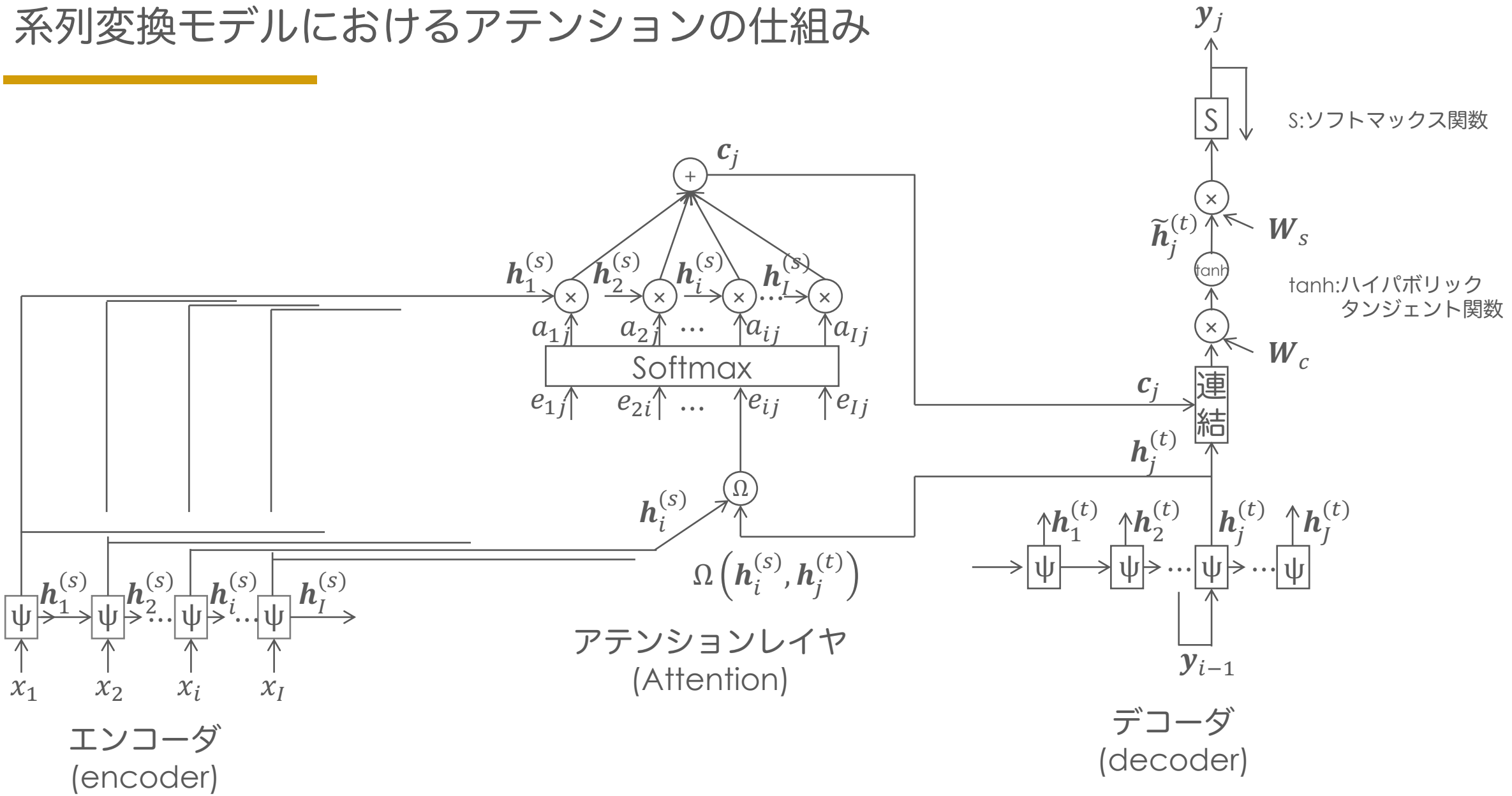
  [;]：結合(concatenate)の記号

- デコーダがj番目の単語を予測する際に、$\widetilde{\boldsymbol{h}}_j^{(t)}$を利用する。

  $p(y_t|y_{<t}, x) = \mathrm{softmax}\left(\boldsymbol{W}_s\ \ \widetilde{\boldsymbol{h}}_j^{(t)}\right)$

- これにより、エンコーダの全ての中間状態をデコーダで利用できることになる。
- このアテンションでは、全て微分可能な関数を用いているため、誤差逆伝播法で勾配を計算することが可能。

系列変換モデルにおけるアテンションの仕組み



アテンションを活用した事例

- アテンションというアイデアは、汎用的なものであり、様々なモデルで取り入れられている
  - 自然言語処理にアテンションを使った事例
    - Google’s Neural Machine Translation System(GNMT)
    - トランスフォーマー
    - 外部メモリを持つニュールラルネットワーク(DAY6で概要を紹介)
- コンピュータビジョン分野では、ビジュアルアテンションという考え方がある
  - 例えば、SENetが参考になる
    - Jie Hu, Li Shen, Samuel Albanie, Gang Sun, Enhua Wu. Squeeze-and-Excitation Networks. CVPR 2018. https://arxiv.org/pdf/1709.01507.pdf

Google’s Neural Machine Translation System

- Google’s Neural Machine Translation System(GNMT)は、Google社が開発したニューラル機械翻訳のシステム
- 実際のサービスとして、2016年から使用されている。
- 論文
  - Yonghui Wu et al. Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation. https://arxiv.org/pdf/1609.08144.pdf
- Google社公式ブログ
  - https://ai.googleblog.com/2016/09/a-neural-network-for-machine.html

ニューラル機械翻訳とは、seq2seqのようなニューラルネットワークモデルを用いて機械翻訳を行うこと

Google’s Neural Machine Translation System

- GNMTは、エンコーダ、デコーダ、アテンションから構成されている。
- GNMTで取り入れられた工夫
  - LSTMレイヤの多層化
  - 双方向LSTM(エンコーダの1層目のみ)
  - スキップコネクション
  - 複数GPUでの分散学習
  - 予測の高速化 (量子化)

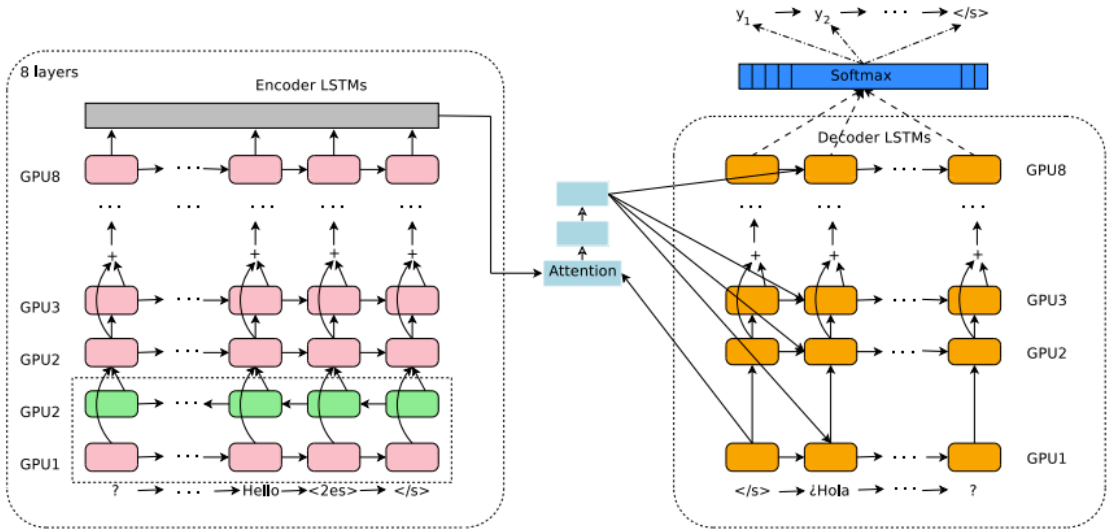
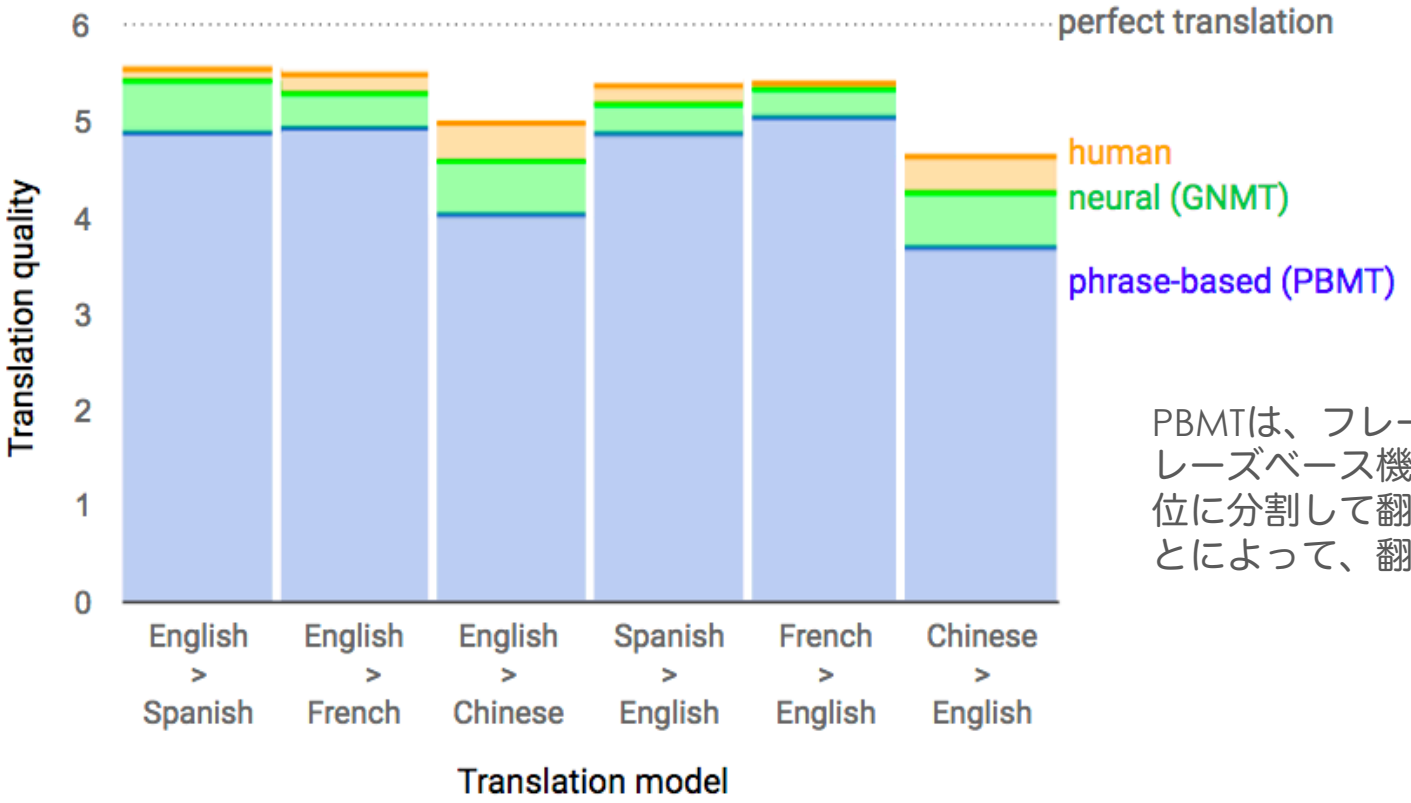


Figure 1: The model architecture of the Multilingual GNMT system. In addition to what is described in [24], our input has an artificial token to indicate the required target language. In this example, the token “<2es>” indicates that the target sentence is in Spanish, and the source sentence is reversed as a processing step. For most of our experiments we also used direct connections between the encoder and decoder although we later found out that the effect of these connections is negligible (however, once you train with those they have to be present for inference as well). The rest of the model architecture is the same as in [24].

Google’s Neural Machine Translation System

- GNMTの精度評価の結果を以下に引用する。

従来のフレーズベース機械翻訳に比べ、人による翻訳に近い精度を実現できていることがわかる



PBMTは、フレーズベース機械翻訳の略。フレーズベース機械翻訳とは、文章を小さな単位に分割して翻訳し、それらを並べ替えることによって、翻訳文を生成する手法。

https://ai.googleblog.com/2016/09/a-neural-network-for-machine.html
より引用

[演習] アテンション付きseq2seqモデルの実装

- 7_4_TimeAttention.ipynb
  - Attentionレイヤとそれを時間方向に束ねるTimeAttentionレイヤを実装しましょう。
- 7_5_AttentionSeq2seq.ipynb
  - アテンション付きseq2seqを計算するためのクラスを実装しましょう。
  - 計算グラフは、「系列変換モデルにおけるアテンションの仕組み」の頁を参考にしましょう。

[演習] 双方向LSTMの実装

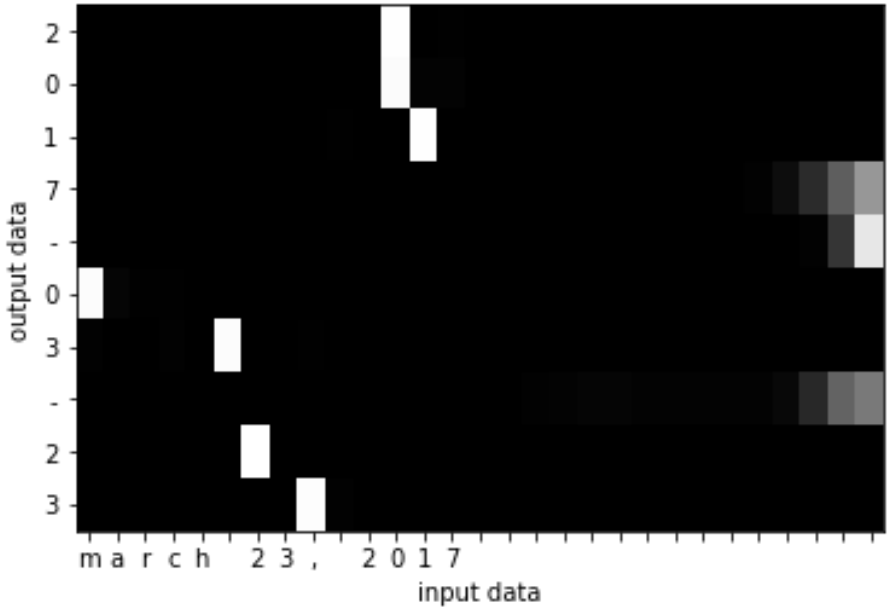
- 7_6_TimeBiLSTM.ipynb
  - 双方向LSTMを計算するためのTimeBiLSTMレイヤを実装しましょう。
- 7_7_AttentionBiSeq2seq.ipynb
  - エンコーダ側を双方向LSTMにしたアテンション付きseq2seqモデルを計算するためのクラスを実装しましょう。
  - 計算グラフは、「系列変換モデルにおけるアテンションの仕組み」の頁を参考にしましょう。

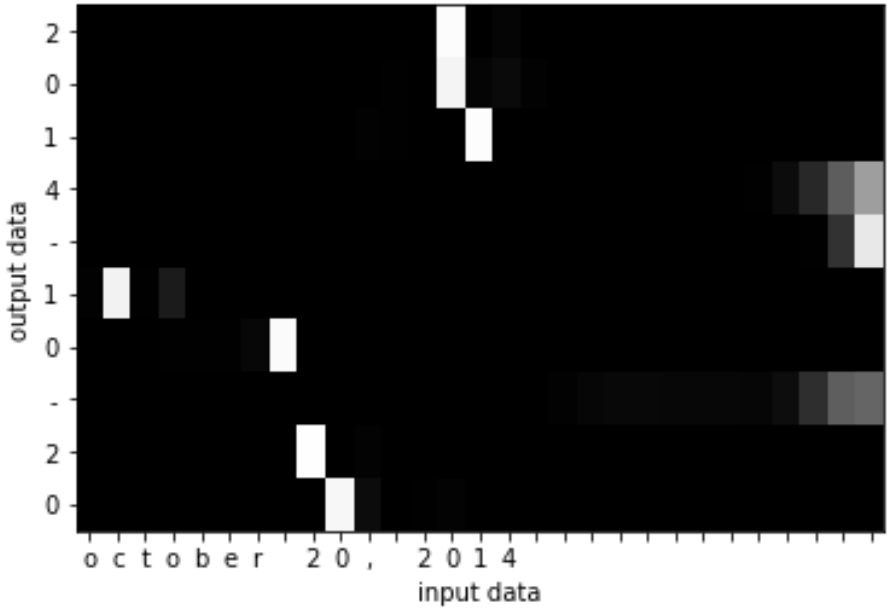
[演習] 系列変換モデルを用いた学習

- 7_8_train_with_seq-to-seq.ipynb
  - 系列変換モデルを用いた学習を行いましょう。
    - seq2seqモデル
    - アテンション付きのseq2seqモデル
    - エンコーダ側を双方向LSTMにしたアテンション付きseq2seqモデル
  - ここでのタスクは、以下のような日付フォーマット変換です。
    - september 27, 1994 -> _1994-09-27
    - August 19, 2003 -> _2003-08-19
    - 2/10/93 -> _1993-02-10
    - 10/31/90 -> _1990-10-31

[演習] アテンションの可視化

- 7_9_visual_attension.ipynb
  - アテンション付きseq2seqの学習済みモデルを用いて、アテンションの結果を可視化しましょう。
  - ある単語を予測する際に、その単語と関連する入力単語に注目(アテンション荷重が大きい)していたら学習がうまくいっています。





白い部分は、アテンション荷重が大きい(1に近い)

[グループワーク] seq2seqモデルの計算グラフ

- Notebook演習で実装した「エンコーダ側を双方向LSTMにしたアテンション付きseq2seqモデル」について、計算グラフを描きましょう。
- 「系列変換モデルにおけるアテンションの仕組み」の頁を参考にしましょう。
- 描くのは、順伝播のみで構いません。
- 2~3名のグループに分かれて、上記課題に取り組みましょう。(20分)
- 最後に、グループごとに発表していただきます。 (15分)

対面講義でこのグループワークを円滑に行うため、
予習の段階で必ずNotebook演習を行なって来てください。

トランスフォーマー

RNNの課題

- これまで、RNNを用いると可変長の時系列データをうまく扱えることを確認してきた。
- しかし、RNNには計算効率面で課題がある。
  - 前時刻に計算した結果を用いながら逐次的に計算を行なっていくため、計算に時間がかかる。
- こういった背景から、RNNを使わずにRNNと同等のことができないかという研究が行われている。
- その一つの成果が、トランスフォーマーである。

トランスフォーマー

- トランスフォーマーは、下記論文で提案された、 RNNを使わずにアテンションのみで機械翻訳問題を解くためのモデル。
- 圧倒的な学習効率で、翻訳タスクにおいてSotA(当時最高性能)を達成。
- これ以降、機械翻訳問題はトランスフォーマーが主流になっていった。
- 原著論文
  - Ashish Vaswani et al. Attention is All You Need, NIPS 2017. https://arxiv.org/pdf/1706.03762.pdf
- この論文では、アテンションに関する明瞭な定式化・解釈を与えている。
  - アテンション＝辞書オブジェクト (Query, Key, Valueモデル)

アテンションの一般化

- アテンンションを辞書(dictionary)オブジェクトとして解釈。
  - 質問文はQuery
  - 事前知識はQueryから検索できるKey
  - 回答候補はKeyと対応するValue
  - KeyとValueのセットがメモリに相当する

$$\text{回答候補} = \text{Softmax}(QK^T) \cdot V$$

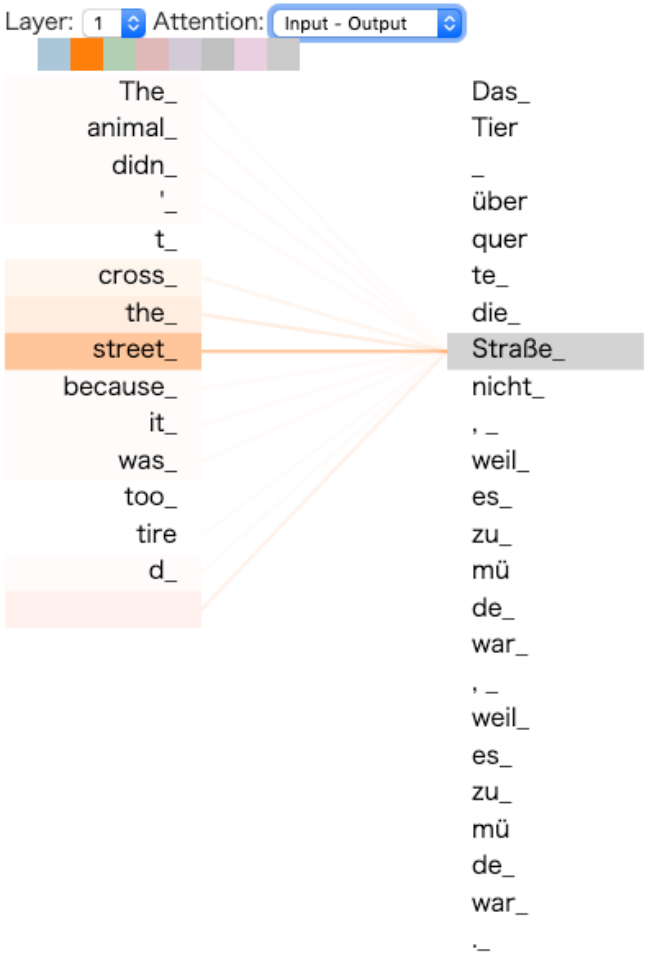
質問文(Query)を入れると、参照すべき場所(Key)が決まり、その場所の回答候補(Value) が得られる。
ソフト・アテンションでは、参照すべき場所(Key)の重みが決まり、その重みに応じて平均することで回答候補を得ていることになる。

ソースターゲット・アテンションとセルフ・アテンション

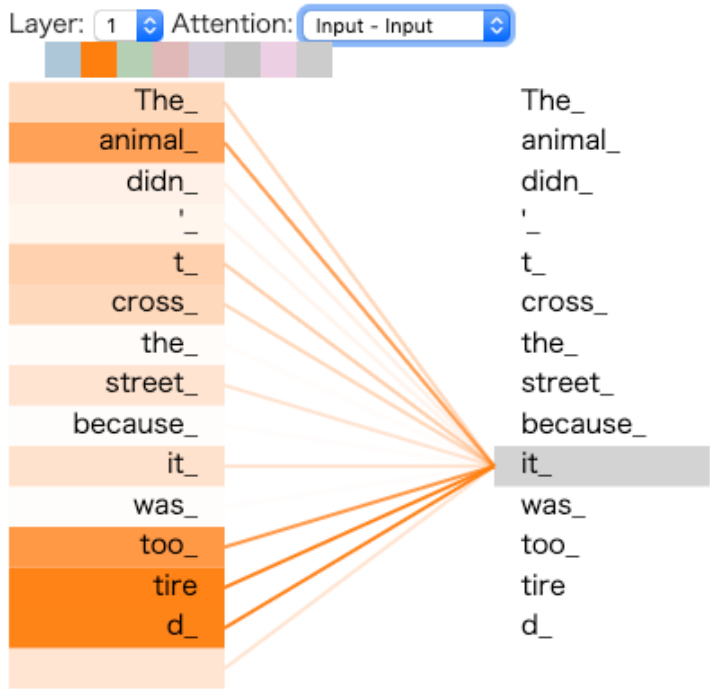
- ソースターゲット・アテンション
  - seq2seqモデルで用いていたアテンション。
  - 2つの系列間の対応関係を捉えることが目的。
  - Queryはデコーダ側、 KeyとValueはエンコーダ側に位置付けられる。
- セルフ・アテンション
  - 文章内の単語間の関係を捉えることが目的。
  - Query、Key、Valueが同じ単語から生成されていく。
  - 各単語が文章中のすべての単語と比較され、その結果が文章内の全ての単語のアテンションスコア(重み)になる。
- トランスフォーマーでは、ソースターゲット・アテンションとセルフ・アテンションの両方が用いられている。

ソースターゲット・アテンションとセルフ・アテンション

ソースターゲット・アテンションの例



セルフ・アテンションの例

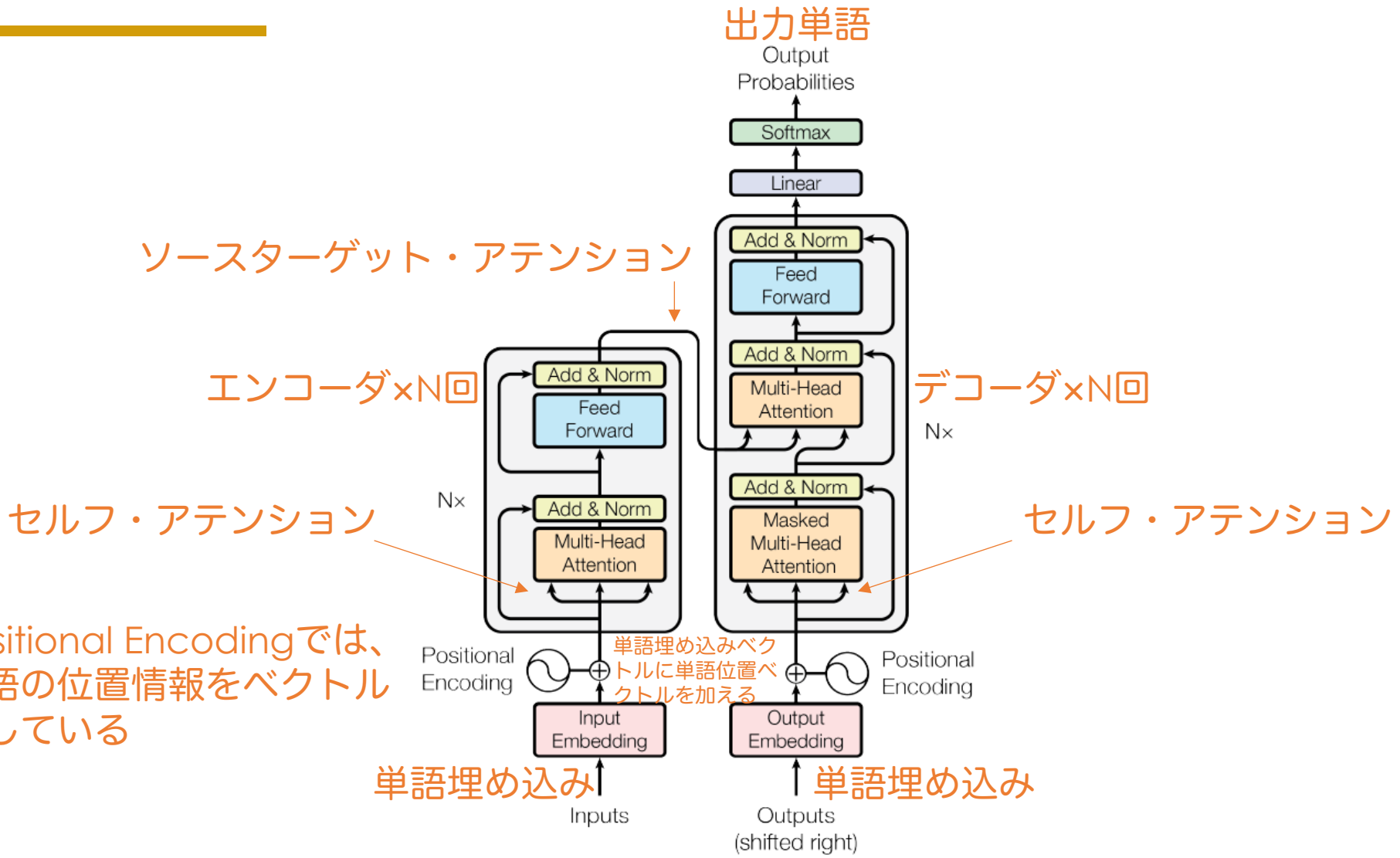


Tensor2Tensor Intro
https://colab.research.google.com/github/tensorflow/tensor2tensor/blob/master/tensor2tensor/notebooks/hello_t2t.ipynb#scrollTo=OJKU36QAfqOC

トランスフォーマーのネットワーク構成

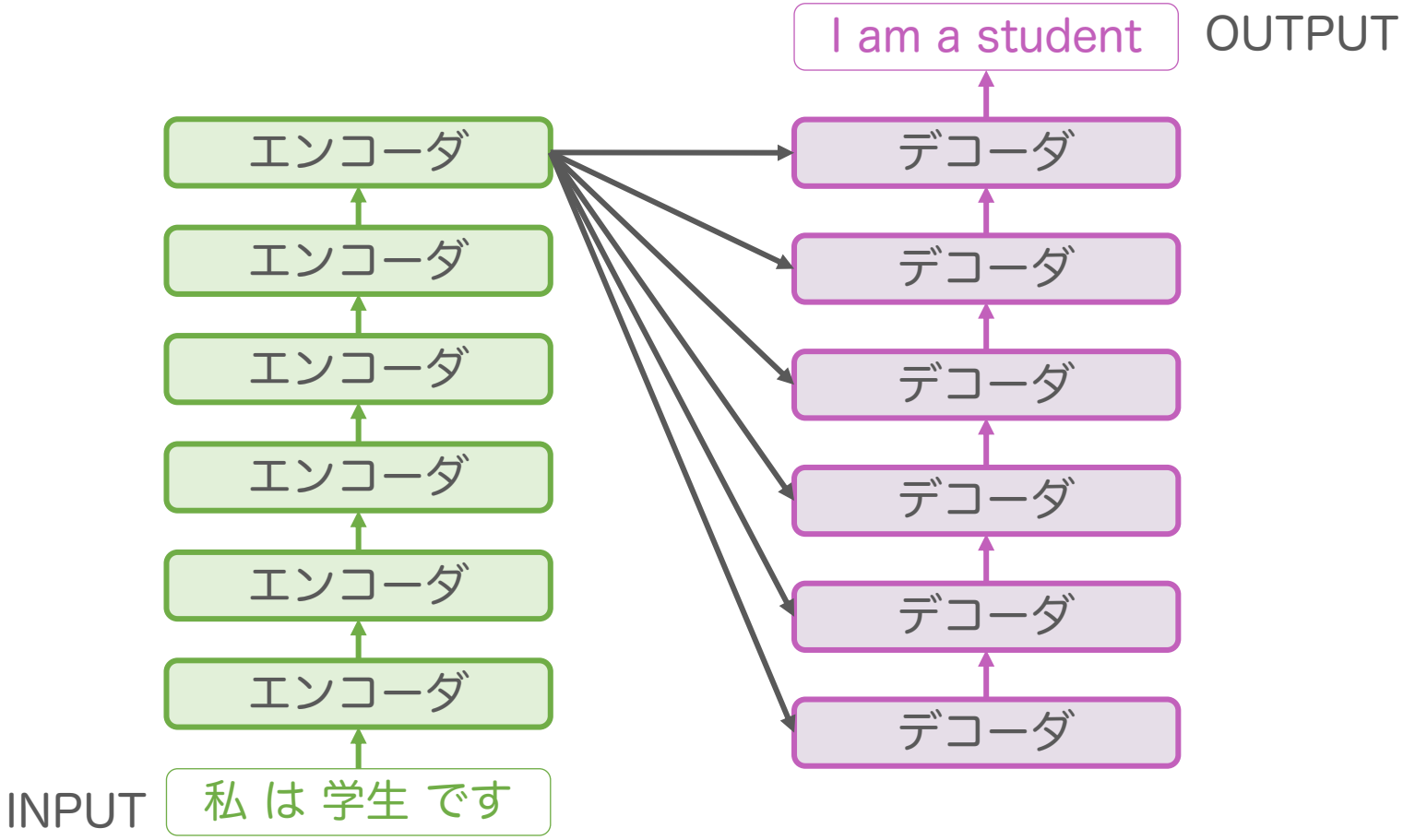
- 次頁からトランスフォーマーのネットワーク構成を見ていく。
- トランスフォーマーの仕組みは、以下のブログが非常に参考になる。
  - http://jalammar.github.io/illustrated-transformer/
- Pythonでの実装については、こちらが参考になる。
  - https://qiita.com/halhorn/items/c91497522be27bde17ce

トランスフォーマーのネットワーク構成



トランスフォーマーのネットワーク構成

- エンコーダおよびデコーダはN回(論文では6回)繰り返される。



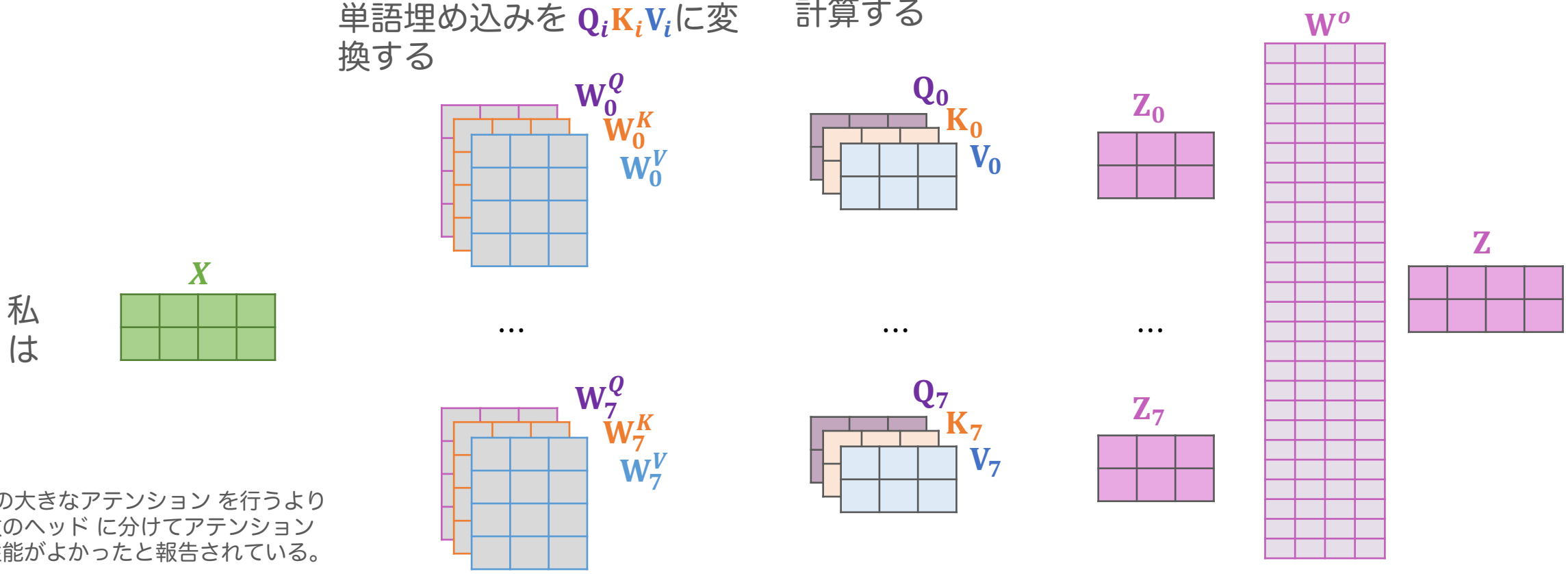
トランスフォーマーのネットワーク構成

$$Z_i = \mathrm{softmax}\left(\frac{Q_i K_i^{T}}{\sqrt{d_k}}\right) V_i$$

d_k: keyベクトルの次元数

- Multi-head attentionの内部構成

1)入力単語　2) 単語を埋め込む　3) ヘッド毎(論文では8ヘッド)にXとW_0 掛けることで、単語埋め込みを $Q_i K_i V_i$に変換する　4) $Q_i K_i V_i$の結果を用いて、Z_iを計算する　5)全てのZ_iを結合し、 W^oと掛ける



論文では、1つの大きなアテンション を行うよりも、小さな複数のヘッド に分けてアテンションを行うほうが性能がよかったと報告されている。

トランスフォーマーのネットワーク構成

- 単語の順番を考慮するために、ポジショナル・エンコーディング(Positional encoding)を導入する

posは、その単語が文章の何番目かを表す数字

d_{model}は、単語埋め込みの次元数

PEは、ポジショナル・エンコーディングの計算結果

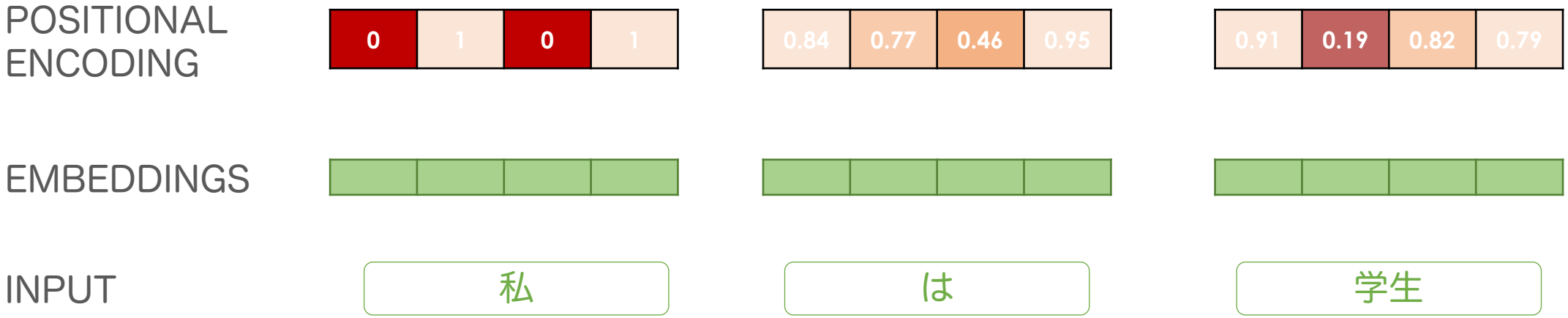
$$PE_{(pos,2i)} = sin(pos/10000^{2i/d_{\text{model}}})$$
$$PE_{(pos,2i+1)} = cos(pos/10000^{2i/d_{\text{model}}})$$

偶数番目の値はsin関数の方を用い、奇数番目の値はcos関数の方を用いる

where pos is the position and i is the dimension. That is, each dimension of the positional encoding corresponds to a sinusoid. The wavelengths form a geometric progression from 2π to $10000 \cdot 2\pi$. We chose this function because we hypothesized it would allow the model to easily learn to attend by relative positions, since for any fixed offset k, PE_{pos+k} can be represented as a linear function of PE_{pos}.

トランスフォーマーのネットワーク構成

- ポジショナル・エンコーディングの例を以下に示す。



参考Notebook: 7_10_Positional_encoding.ipynb

BERT

- BERTとは、以下の論文で提案された自然言語処理用モデル。
  - Bidirectional Encoders’ Representations from Transformer (BERT)
- 双方向トランスフォーマーを用いている。
- Jacob Devlin et al. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. (Arxiv preprint, from Google, Oct. 2017). https://arxiv.org/pdf/1810.04805.pdf

BERT

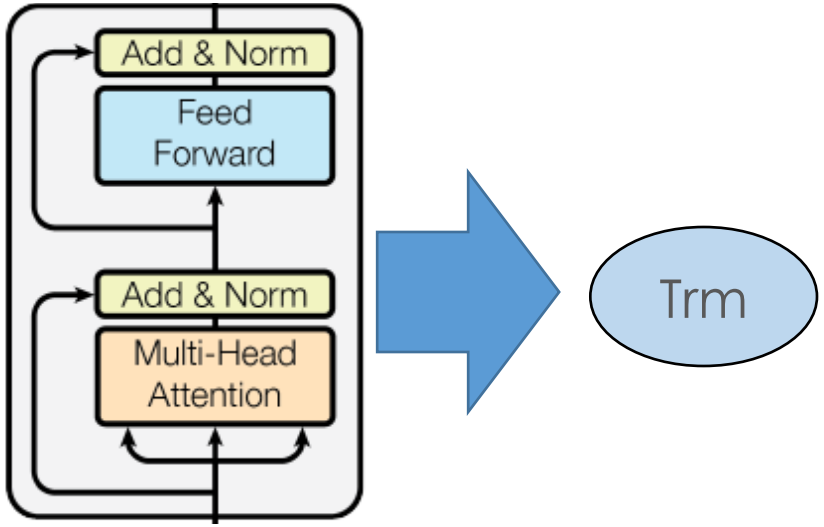
- 自然言語処理には、翻訳・質疑応答・素性分析etc…と様々なタスクが存在している。
  - タスクごとに有用な言語の特徴表現は別々に設計されており、万能の特徴表現がない。
- BERTはあらゆるタスクにおいて有用な表現学習を行えるモデルであり、その学習則の提案でもある。
- 感情分析・含意関係認識・質疑応答・意味等価性判別など、11のタスクでSotA(当時最高性能) 。
  - 後に提案された以下の論文では翻訳タスクでもStoA。
    - Guillaume Lample, Alexis Conneau. Cross-lingual Language Model Pretraining. https://arxiv.org/pdf/1901.07291.pdf
- 自然言語処理の決定的手法になるかもしれないと話題の手法

BERT

- BERTでは、事前学習により、言語表現(分散表現)を獲得する。
  - word2vecよりも豊かな表現を得ることができる。
- 事前学習済みのモデルに、層を追加することによって、特定のタスクを解けるようにする。

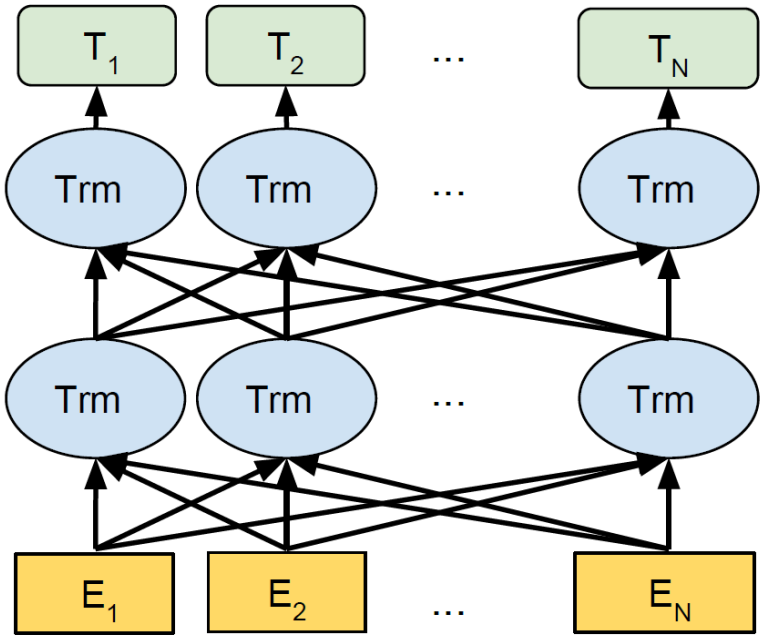
双方向トランスフォーマー

- トランスフォーマーのエンコーダ側を重み付き線形和で重ねる。
- (E_i)は入力系列。例、 “He” “bought” “an” “apple”
- 入力系列に対して特徴表現の系列(T_i)が出力される。



トランスフォーマーの
エンコーダ側ブロック

(原論文 Fig. 1から引用・一部改変)



双方向トランスフォーマー

BERTにおける事前学習

- 双方向トランスフォーマー自体は非常にシンプル
  - 万能の言語処理機とするために「何を学習させるか」が重要

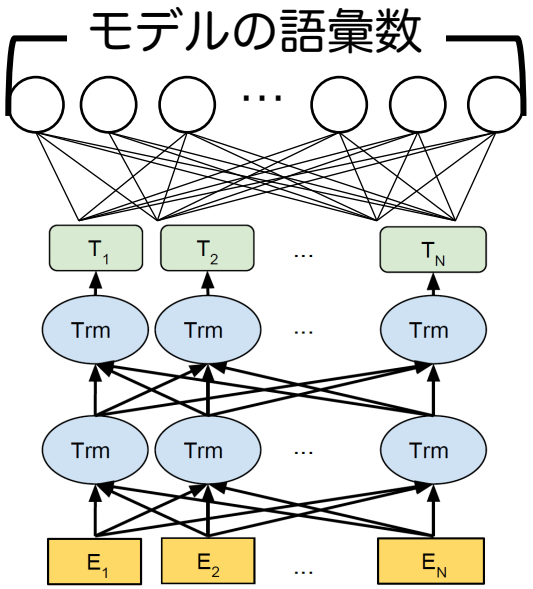
① 単語マスク問題: 入力系列のうち、隠された単語がなにかを予測(局所的な特徴学習)

② 隣接文問題: 2つの入力文が隣り合うものかどうかを判別(大域的な特徴学習)

①

Q.
We went to Tokyo
across the Pacific Ocean
by _____.

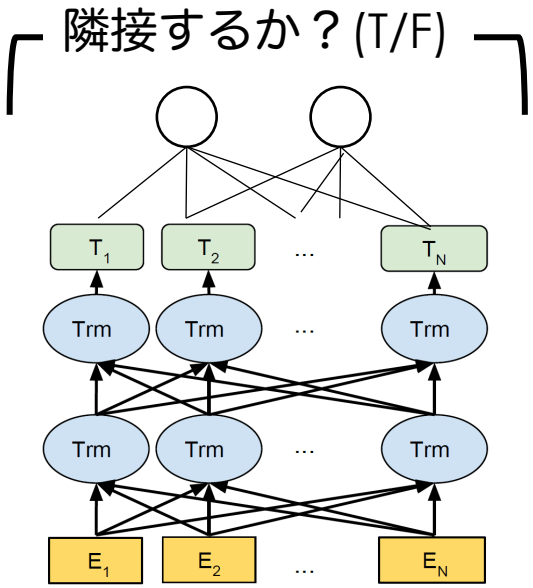
A.
Kayac



②

1. I have a pen.
2. I have an apple.
   => True!

1. I know deep learning.
2. Who stole a branket?
   => False!



外部メモリを持つニューラルネットワーク

外部メモリを持つニューラルネットワーク

- 外部メモリを持つニューラルネットワークについては、DAY6で概要を紹介した。
- ここでは、外部メモリを持つニューラルネットワークをアテンションの応用例として、より詳しい内容を紹介する。

Memory Networks

- Memory Networksは、記憶とそれを呼び起こす仕組みの一般的なモデル
- どのような構成で質疑応答システムに適したモデルが作れるかを提案している
- Memory Networksの各コンポーネントにニューラルネットワークを用いたものを Memory Neural Networks(MemNN)という。
- 原著論文
  - Jason Weston, Sumit Chopra, Antoine Bordes. Memory Networks. ICLR 2015. https://arxiv.org/pdf/1410.3916.pdf

Memory Networks

- 質疑応答システムをDNNで実現する方法を考える。
- 質疑応答では事前に与えられた知識を何らかの形で蓄えておかなければならない。

事前知識=学習データ (事実; fact)

机の上には美味しそうないちごとバナナが置いてあります。
太郎君はバナナを食べたそうにしています。
でも、太郎君はお母さんに止められているのでバナナを食べられません。
なぜなら太郎君は先週の健康診断で体重が増えてしまっていたからです。

質問(入力文)

Q. 机の上にはバナナの他に何がおいてありますか？

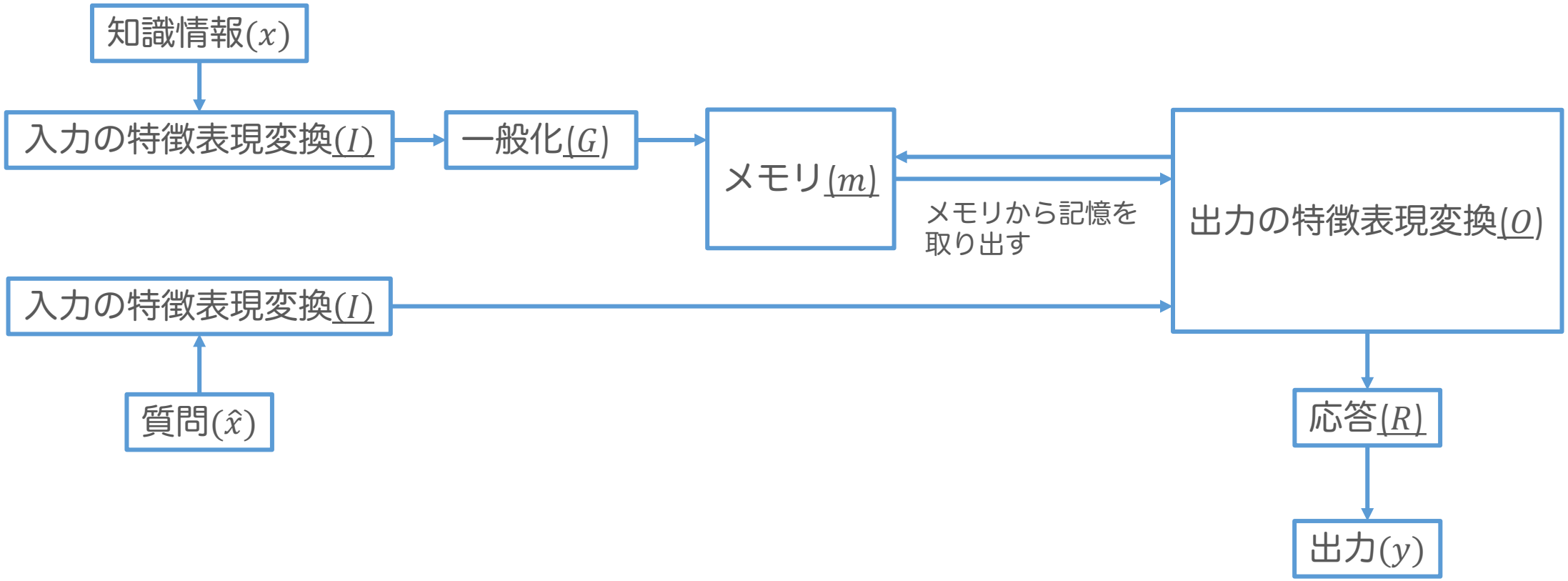
回答(NNの出力)

A. いちご

Memory Networks概観

- MemNNの主な構成要素は5つ
  - メモリ(m): 知識(入力された文、あるいはその特徴)を蓄えておく配列
  - 入力の特徴表現変換(I): 入力された知識を特徴表現に変換する部分
  - 入力の一般化(G): 新たな入力を用いてメモリMを更新する
  - 出力の特徴表現変換(O): 質問から回答に対応する特徴表現を作成する部分
  - 応答(R): Oを用いて実際に文としての回答を生成する部分
- 知識情報: 入力の知識文x
  - 特徴表現に変換 $h = I(x)$, 知識としてメモリに格納: $m_i = h$
- 質問: 入力の質問文 $\hat{x}$
  - 質問文を特徴表現に変換 $\hat{h} = I(\hat{x})$, メモリから知識を引いて回答 $y = R(O(\hat{h}, m))$

Memory Networksの構成要素



I, G, O, Rモジュール

- I(Input)モジュールでは生の文字列を特徴表現に変換する。
  - Bag of Words, Word2Vecなど、なんでもよい
  - このIによってモデルが行う文章の表現が定まる
- G(Generalize)モジュールでは知識をメモリmに格納する。
  - mは特徴表現$I(x)$を値に持つただの配列 $m = (m_i)_{i \in I}$
  - Gモジュールはmの格納位置を決定関数$H(x)$で決めて代入(保存)
    - $m_{H(x)} = I(x)$
    - 最も単純には後ろに追加していくだけの格納:

$$m_{H(x)} = I(x) \ \ where \ \ H(x) = N, N = N + 1$$

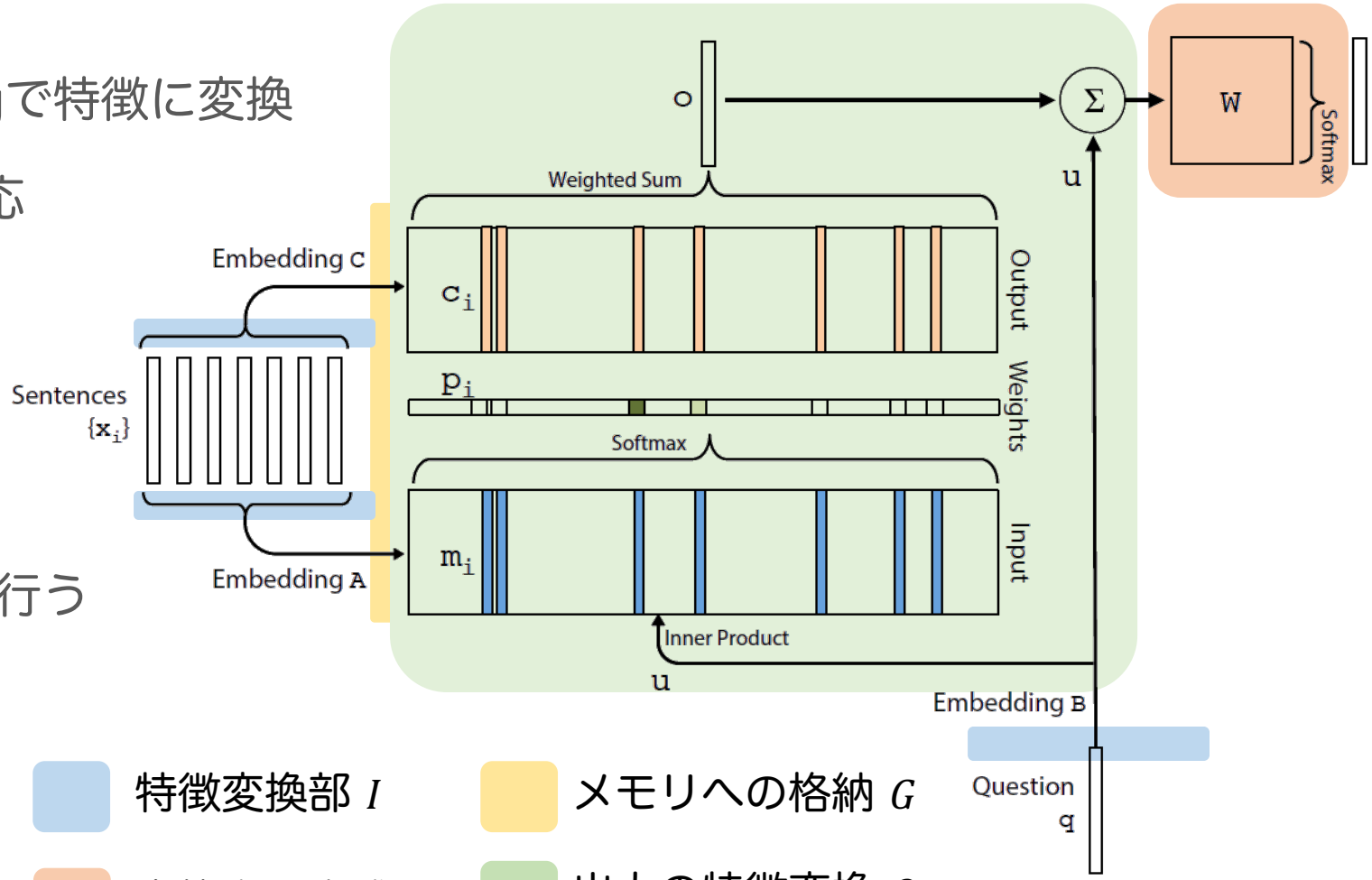
- Oモジュールは格納された知識の中から、上位k個の回答に役立つメモリm_iを見つけてくる。
- Rモジュールは求められた$O(x)$から回答文rを生成する

End-To-End Memory Networks

- Memory Networks(MemNN)の考え方を微分可能なニューラルネットワークで実現したモデル
  - End-to-End MemNN => **MemN2N** と略される
- MemNNでは、ハード・アテンションでメモリから記憶を取り出すことを考えていた。
  - MemN2Nでは、ソフト・アテンションを用いる。これにより、モデル全体を1つのニューラルネットワークで構築することが可能になった。
- 原著論文
  - End-To-End Memory Networks. Sainbayar Sukhbaatar et al. NIPS 2015. https://arxiv.org/pdf/1503.08895.pdf

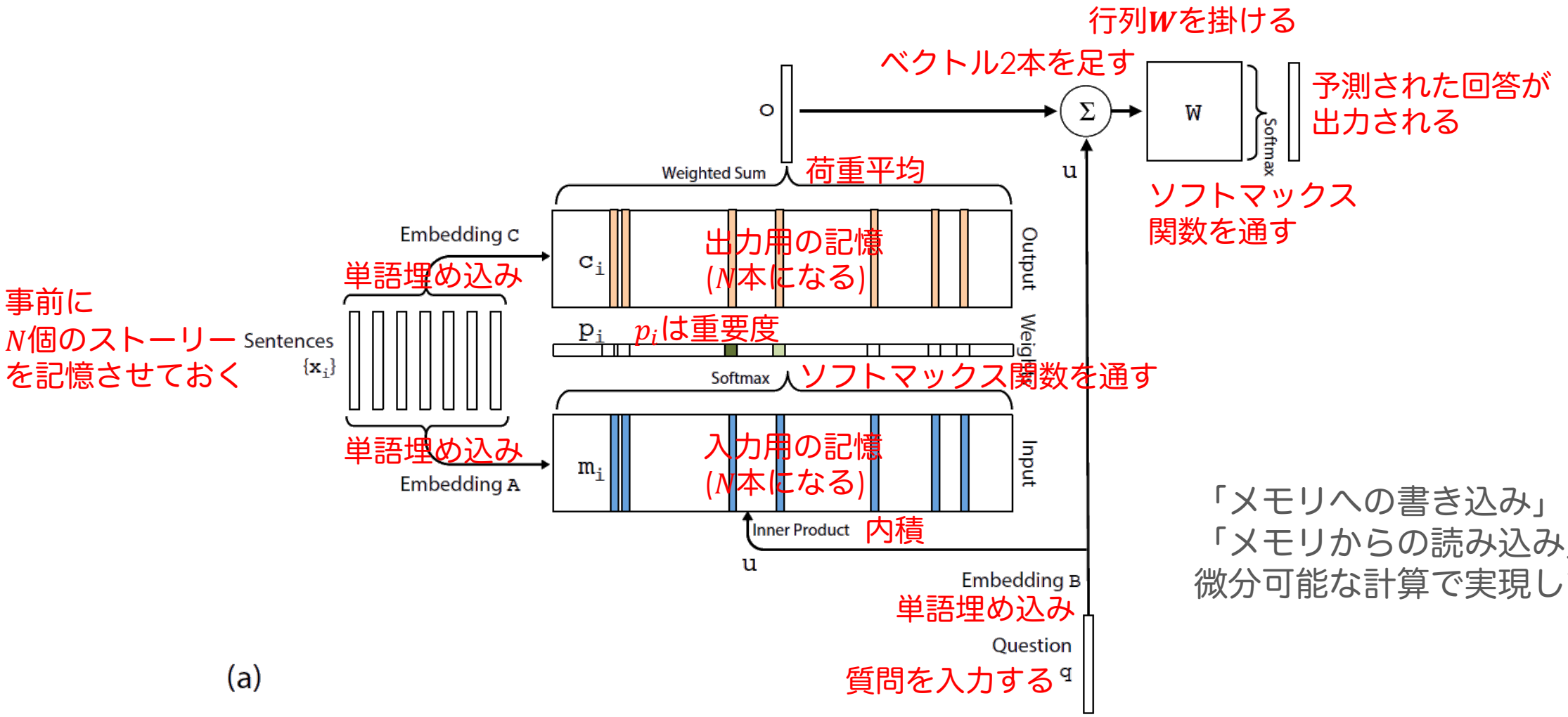
MemN2N概観

- 入力された文はWord embeddingで特徴に変換
  - MemNNの特徴変換 $I(\boldsymbol{x})$に対応
- MemN2Nでは入力用と出力用で異なる表現のメモリを持つ
  - 入力用$\boldsymbol{m_i}$,出力用 $\boldsymbol{c_i}$
  - それぞれ異なるembeddingを行う
- 出力文を生成するOモジュールはソフト・アテンションを行う
- 生成されたOと質問文の表現B(x)から回答を生成する



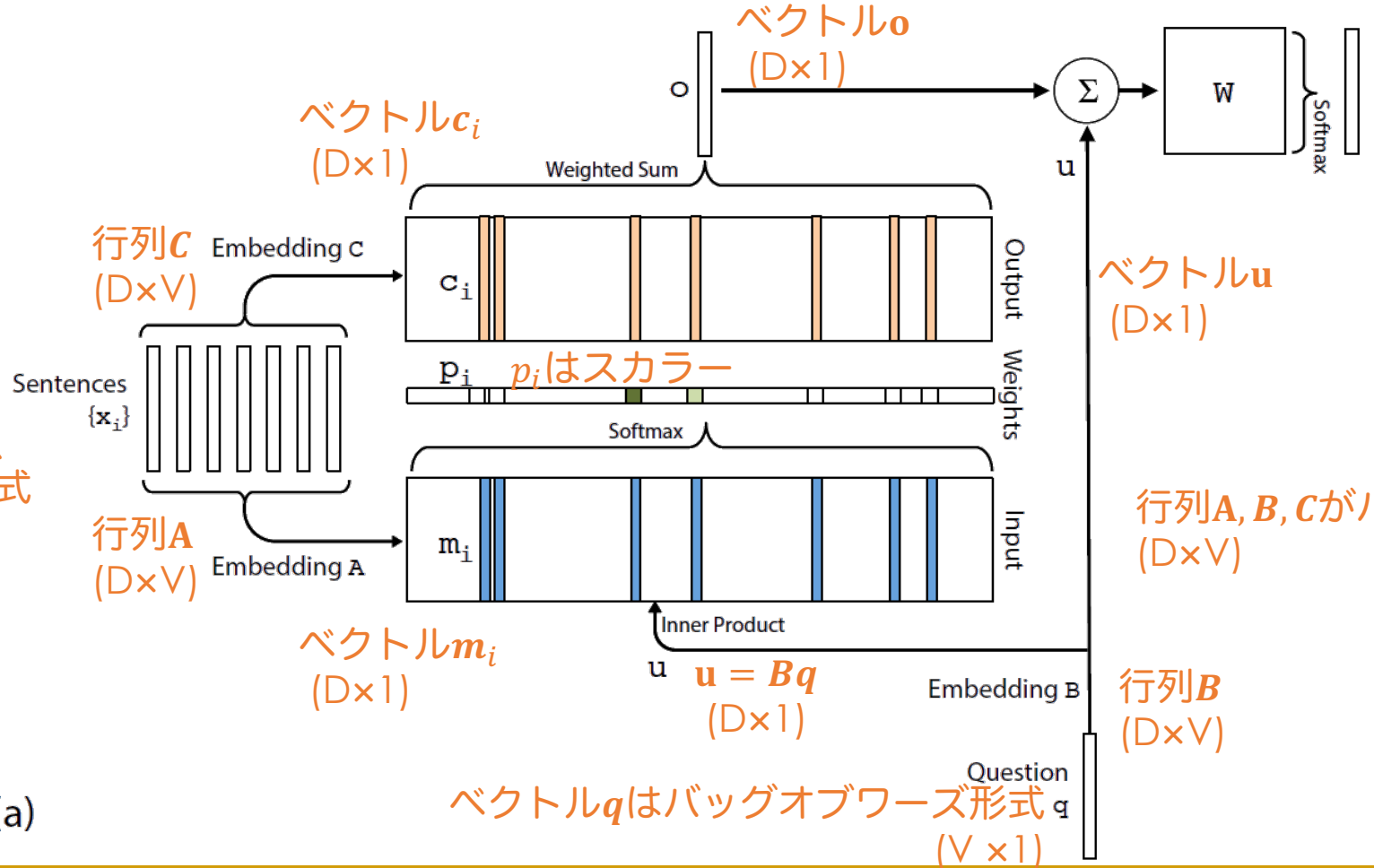
(原論文 Fig. 1から引用・一部改変)

MemN2Nの構成



「メモリへの書き込み」と
「メモリからの読み込み」を
微分可能な計算で実現している

MemN2Nの構成



ベクトル$\mathbf{o}$
(D×1)

1つの単語が出力される

ベクトル$\boldsymbol{c}_i$
(D×1)

行列$\boldsymbol{C}$
(D×V)

ベクトル$\mathbf{u}$
(D×1)

p_iはスカラー

あるストーリー$\boldsymbol{x}_i$は、
バッグオブワーズ形式
(V ×1)

行列$\mathbf{A}, \boldsymbol{B}, \boldsymbol{C}$がパラメータ(単語埋め込み行列)
(D×V)

行列$\mathbf{A}$
(D×V)

ベクトル$\boldsymbol{m}_i$
(D×1)

$\mathbf{u} = \boldsymbol{B}\boldsymbol{q}$
(D×1)

行列$\boldsymbol{B}$
(D×V)

ベクトル$\boldsymbol{q}$はバッグオブワーズ形式
(V ×1)

その他の話題

CNNとRNNの組み合わせ：画像キャプション生成タスク

- 画像の内容を自然言語で記述するタスク。
- 画像キャプション生成モデルの例として、Neural Image Captionがある。
  - Oriol Vinyals, Alexander Toshev, Samy Bengio, Dumitru Erhan. Show and Tell: A Neural Image Caption Generator. https://arxiv.org/pdf/1411.4555.pdf

CNN
(GoogLeNet)

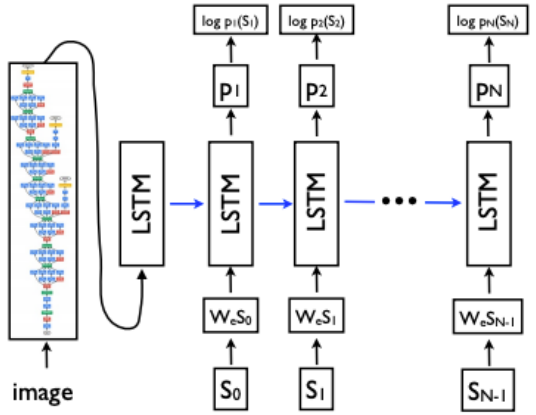


Figure 3. LSTM model combined with a CNN image embedder (as defined in [12]) and word embeddings. The unrolled connections between the LSTM memories are in blue and they correspond to the recurrent connections in Figure 2. All LSTMs share the same parameters.

Figure 5. A selection of evaluation results, grouped by human rating.

CNNとRNNの組み合わせ：画像質問応答タスク

- 画像に関する質問に答えるタスク。
- CNNとRNNを組み合わせたモデルが利用される。
- VQA用データセットが提供されているHP。
  - https://visualqa.org/

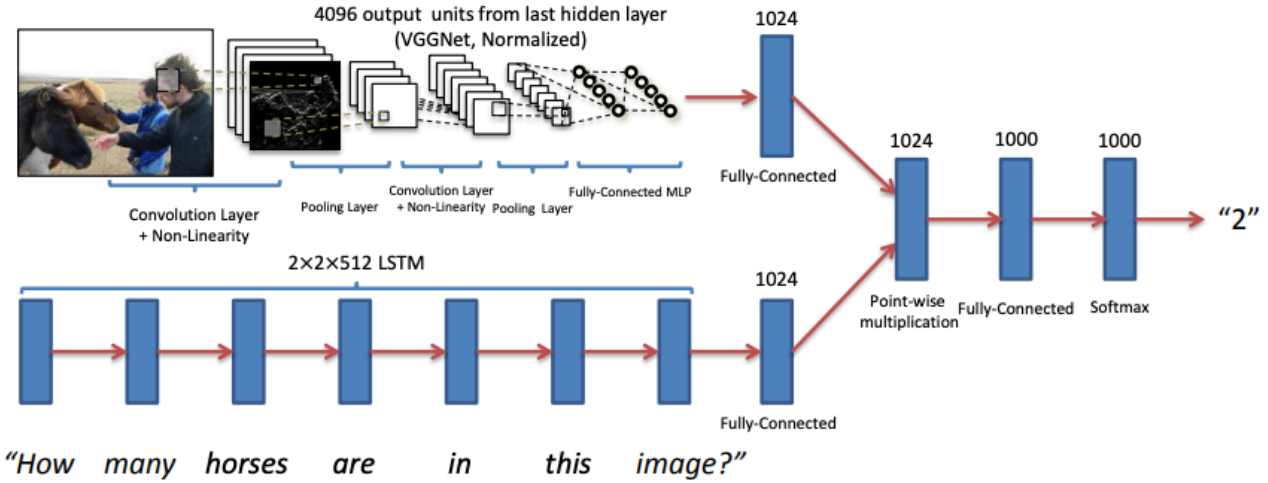


Fig. 8: Our best performing model (deeper LSTM Q + norm I). This model uses a two layer LSTM to encode the questions and the last hidden layer of VGGNet [48] to encode the images. The image features are then ℓ_2 normalized. Both the question and image features are transformed to a common space and fused via element-wise multiplication, which is then passed through a fully connected layer followed by a softmax layer to obtain a distribution over answers.

Fig. 1: Examples of free-form, open-ended questions collected for images via Amazon Mechanical Turk. Note that commonsense knowledge is needed along with a visual understanding of the scene to answer many questions.

Aishwarya Agrawal∗ , Jiasen Lu∗ , Stanislaw Antol∗ , Margaret Mitchell, C. Lawrence Zitnick, Dhruv Batra, Devi Parikh.
VQA: Visual Question Answering.
https://arxiv.org/pdf/1505.00468.pdf

講座の時間が余ったら

- 今回の復習をします。
- 次回の予習をします。