

Three green apples are arranged in a cluster. One apple is in the foreground, slightly to the right, and two are behind it, one to the left and one to the right. They are all bright green with some yellowing at the stems.

人工知能

IE229 – ARTIFICIAL INTELLIGENCE

第14回講義: 自然言語処理 (3)

Lecture 14: Natural Language Processing (3)

二宮 崇 (Takashi Ninomiya)

愛媛大学 (Ehime University)

ninomiya@cs.ehime-u.ac.jp

人工知能第14回の講義です。第14回は機械翻訳の続きを学びます。

今回の講義内容

- **自然言語処理 (Natural Language Processing)**
 - ニューラル機械翻訳 (Neural Machine Translation)
 - アテンション付きエンコーダー・デコーダーモデル
 - Transformer



2

今回はニューラル機械翻訳の中でも有名なアテンション付きエンコーダー・デコーダーモデルとTransformerモデルについて学びます。

アテンション付きエンコー
ダー・デコーダーモデル
(ENCODER-DECODER MODEL
WITH ATTENTION)

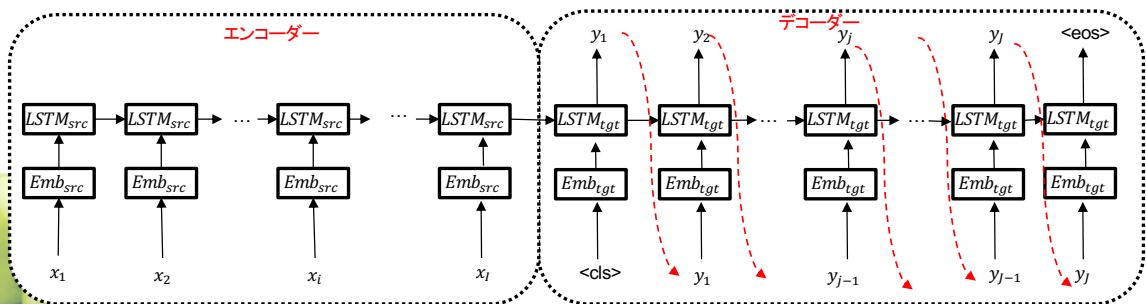


LSTMに基づく エンコーダー・デコーダーモデル

● LSTMに基づくエンコーダー・デコーダーモデル

- 入力文: $x_1, x_2, \dots, x_i, \dots, x_I$ 、出力文: $y_1, y_2, \dots, y_j, \dots, y_J$
- エンコーダー: $h_i = LSTM_{src}(Emb_{src}(x_i), h_{i-1})$
- デコーダー: $h_j = LSTM_{tgt}(Emb_{tgt}(y_{j-1}), h_{j-1})$
 $y_j = softmax(W_o h_j + b_o)$

- **しかし、LSTMを使っても、エンコーダーとデコーダーの間でかなりの距離があるので情報がうまく伝播されない**

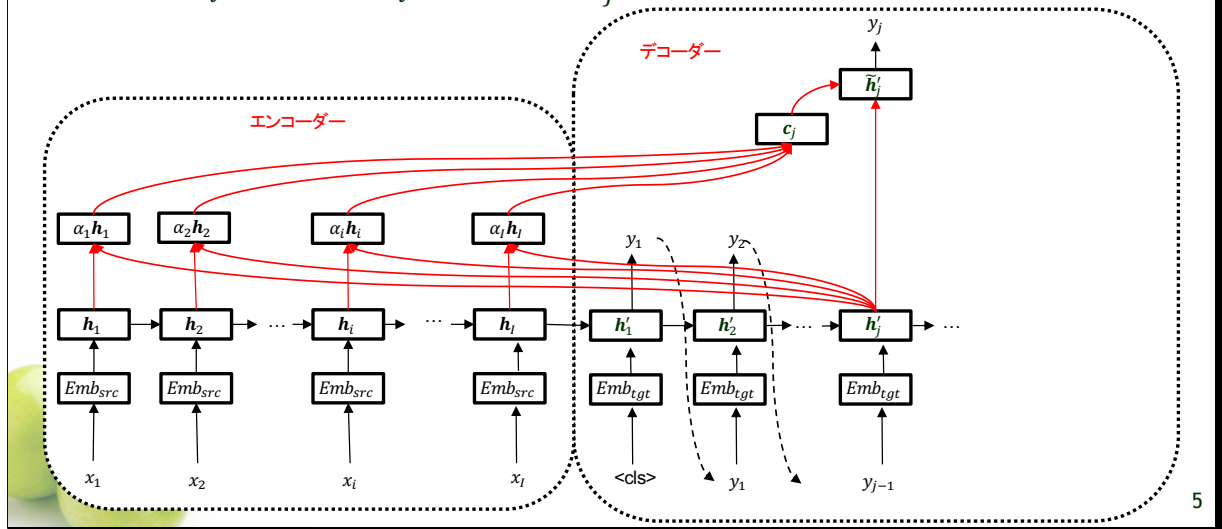


4

LSTMに基づくエンコーダー・デコーダーモデルの概要図です。LSTMによってRNNに基づくエンコーダー・デコーダーモデルがかなり改善されることはわかりましたが、LSTMを使っても、エンコーダーとデコーダーの間でかなり距離があるため、情報がうまく伝播されないという問題があります。

アテンション付き エンコーダー・デコーダーモデル

- デコーダーの出力の際にエンコーダーの内部状態を直接参照することで長距離の問題を解決する
- デコーダーの内部状態(h'_j)に対するエンコーダーの内部状態(h_i)の重み (α_i)を求め、 h_i の加重平均(c_j ; コンテキストベクトル)を求める



そこで、新しく提案されたモデルが、アテンション付きエンコーダー・デコーダーモデルです。

デコーダーの出力の際にエンコーダーの内部状態を直接参照することで長距離の問題を解決します。このスライドの図がモデル全体の概要になりますが、みたとおり、図で書くとかなり複雑な仕組みになっているようにみえます。実際に数式で表現するとかなりシンプルな形で書けるため、アテンション付きエンコーダー・デコーダーモデルは数式で記述されることが多いです。

エンコーダーの処理は今までと同じなのですが、デコーダーの動きが異なります。ある時刻 j の処理をみてみましょう。デコーダーの内部状態(h'_j)に注目します。 h'_j に対するエンコーダーの内部状態(h_i)の重み (α_i)を求め、 h_i の加重平均を計算することで、 h'_j に対するエンコーダー全体の状態をコンテキストベクトル(c_j)の形で計算します。最後にコンテキストベクトル(c_j)と h'_j から新しい内部状態 \tilde{h}'_j を求めて、 y_j を出力します。

アテンション付き エンコーダー・デコーダーモデル

● アテンション付きエンコーダー・デコーダーモデルの形式化

- 入力文: $x_1, \dots, x_i, \dots, x_I$
- 出力文: $y_1, \dots, y_j, \dots, y_J$
- エンコーダー: $\mathbf{h}_i = LSTM_{src}(Emb_{src}(x_i), \mathbf{h}_{i-1})$ ($\mathbf{h}_i \in \mathbb{R}^d$)
- デコーダー: $\mathbf{h}'_j = LSTM_{tgt}(Emb_{tgt}(y_{j-1}), \mathbf{h}'_{j-1})$ ($\mathbf{h}'_j \in \mathbb{R}^d$)

アテンションの重み $\alpha_i = \frac{\exp(\mathbf{h}_i^T \mathbf{h}'_j)}{\sum_{i'=1}^I \exp(\mathbf{h}_{i'}^T \mathbf{h}'_j)}$ ($\alpha_i \in \mathbb{R}$)

コンテキストベクトル $\mathbf{c}_j = \sum_{i=1}^I \alpha_i \mathbf{h}_i$ ($\mathbf{c}_j \in \mathbb{R}^d$)

$$\tilde{\mathbf{h}}'_j = \tanh(W_c[\mathbf{c}_j; \mathbf{h}'_j] + \mathbf{b}_c) \quad (W_c \in \mathbb{R}^{d \times 2d})$$

$$y_j = \text{softmax}(W_o \tilde{\mathbf{h}}'_j + \mathbf{b}_o) \quad (W_o \in \mathbb{R}^{V \times d})$$



;は連結、 d は内部状態の次元数、 V は語彙数

6

アテンション付きエンコーダー・デコーダーモデルを定式化するとこのようになります。

入力文: $x_1, \dots, x_i, \dots, x_I$

出力文: $y_1, \dots, y_j, \dots, y_J$

エンコーダー: $\mathbf{h}_i = LSTM_{src}(Emb_{src}(x_i), \mathbf{h}_{i-1})$

とします。エンコーダーはただのエンコーダー・デコーダーモデルと同じです。ただし、エンコーダーの各時刻における内部状態を全て記録しておき、それらにアクセスできるようにしておきます。

デコーダー次のように定義されます。まず、普通のエンコーダー・デコーダーモデルと同じ様に内部状態 \mathbf{h}'_j を計算します。

$$\mathbf{h}'_j = LSTM_{tgt}(Emb_{tgt}(y_{j-1}), \mathbf{h}'_{j-1})$$

続いて、エンコーダーの内部状態の系列 $(\mathbf{h}_1, \dots, \mathbf{h}_i, \dots, \mathbf{h}_I)$ に対して、それらの重みを計算します。

$$\alpha_i = \frac{\exp(\mathbf{h}_i^T \mathbf{h}'_j)}{\sum_{i'=1}^I \exp(\mathbf{h}_{i'}^T \mathbf{h}'_j)}$$

重みは、単純にデコーダーの内部状態 \mathbf{h}'_j とエンコーダーの内部状態 \mathbf{h}_i との内積に対して、softmaxの計算を行っているだけ、ということに注意してください。

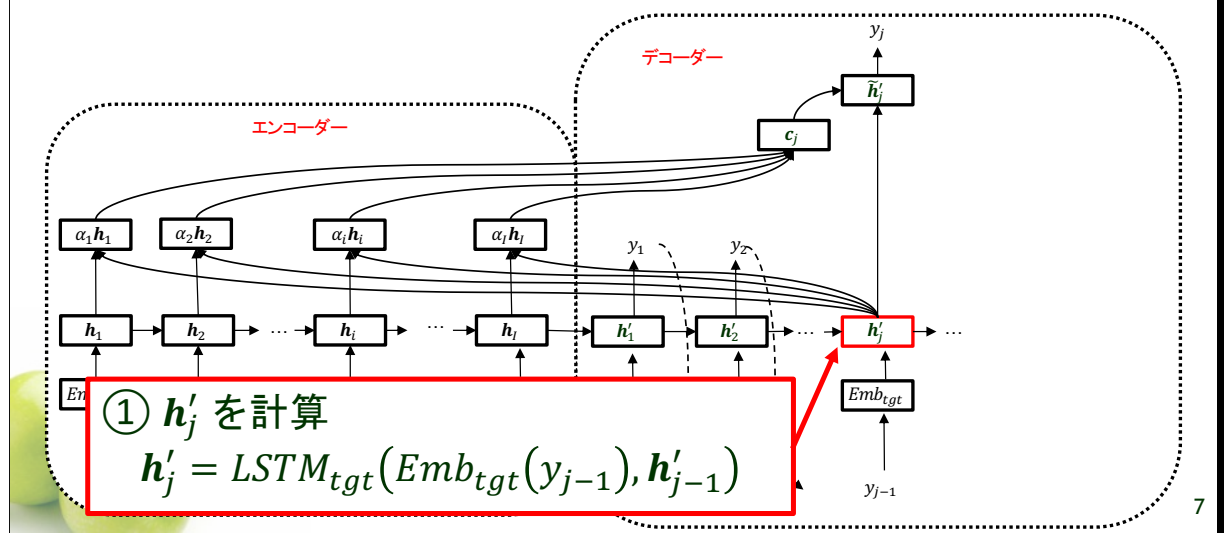
コンテキストベクトル \mathbf{c}_j はエンコーダーの内部状態の系列 $(\mathbf{h}_1, \dots, \mathbf{h}_i, \dots, \mathbf{h}_I)$ に対する加重平均で求められます。

$$\mathbf{c}_j = \sum_{i=1}^I \alpha_i \mathbf{h}_i$$

最後にコンテキストベクトル \mathbf{c}_j とデコーダーの内部状態 \mathbf{h}'_j を連結したベクトルに

対して、線形変換とハイパボリックタンジェントをかけて、新しい内部状態 \tilde{h}_j を得ます。この内部状態に線形変換とsoftmaxを適用して、単語出力の確率分布を得ます。

アテンション付き エンコーダー・デコーダーモデル



概要図をみながら順にみていきましょう。

最初にLSTMデコーダーにより内部状態 h'_j を計算します。

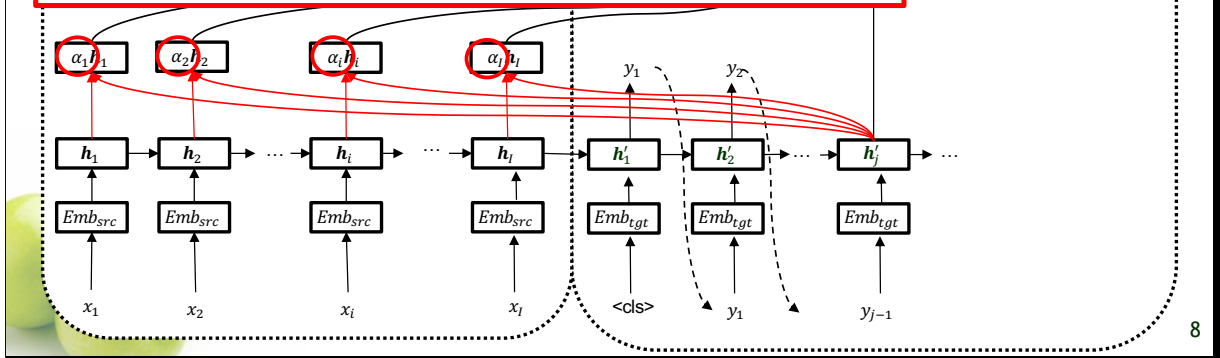
$$h'_j = LSTM_{tgt}(Emb_{tgt}(y_{j-1}), h'_{j-1})$$

アテンション付き エンコーダー・デコーダーモデル

② h_i と h'_j から α_i を計算($1 \leq i \leq I$)

$$\alpha_i = \frac{\exp(h_i^T h'_j)}{\sum_{i'=1}^I \exp(h_{i'}^T h'_j)}$$

- α_i は h_i と h'_j の内積にsoftmaxをかけたもの
- $0 \leq \alpha_i \leq 1, \sum_{i=1}^I \alpha_i = 1$ となることに注意



続いて、 h_i と h'_j から α_i を計算します($1 \leq i \leq I$)。

$$\alpha_i = \frac{\exp(h_i^T h'_j)}{\sum_{i'=1}^I \exp(h_{i'}^T h'_j)}$$

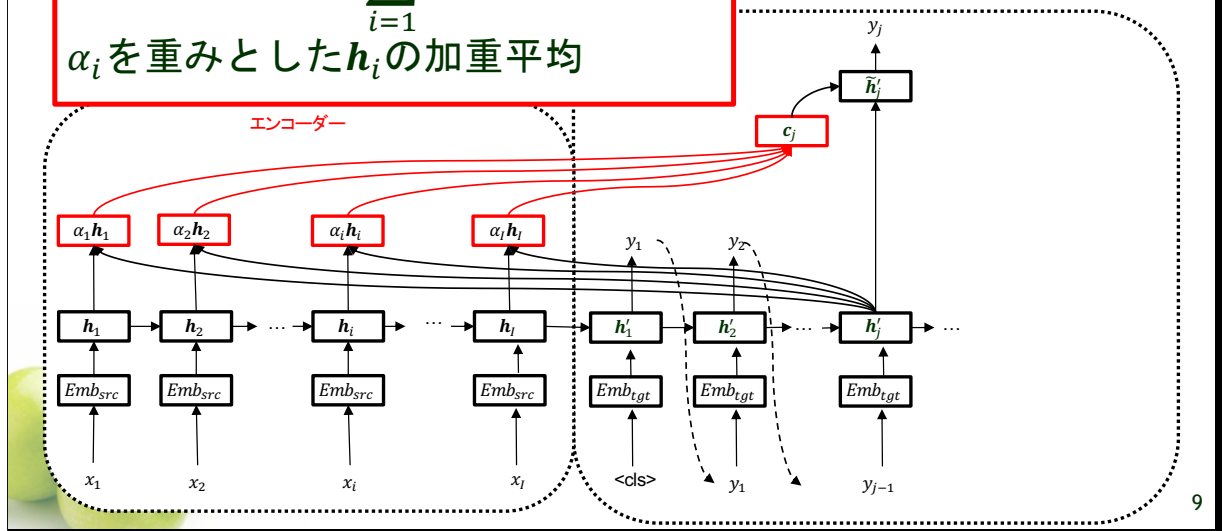
α_i は h_i と h'_j の内積にsoftmaxをかけたものであるということ、 $0 \leq \alpha_i \leq 1, \sum_{i=1}^I \alpha_i = 1$ となることに注意してください。

アテンション付き エンコーダー・デコーダーモデル

③ コンテキストベクトル c_j を計算

$$c_j = \sum_{i=1}^I \alpha_i h_i$$

α_i を重みとした h_i の加重平均



コンテキストベクトル c_j を計算します。

$$c_j = \sum_{i=1}^I \alpha_i h_i$$

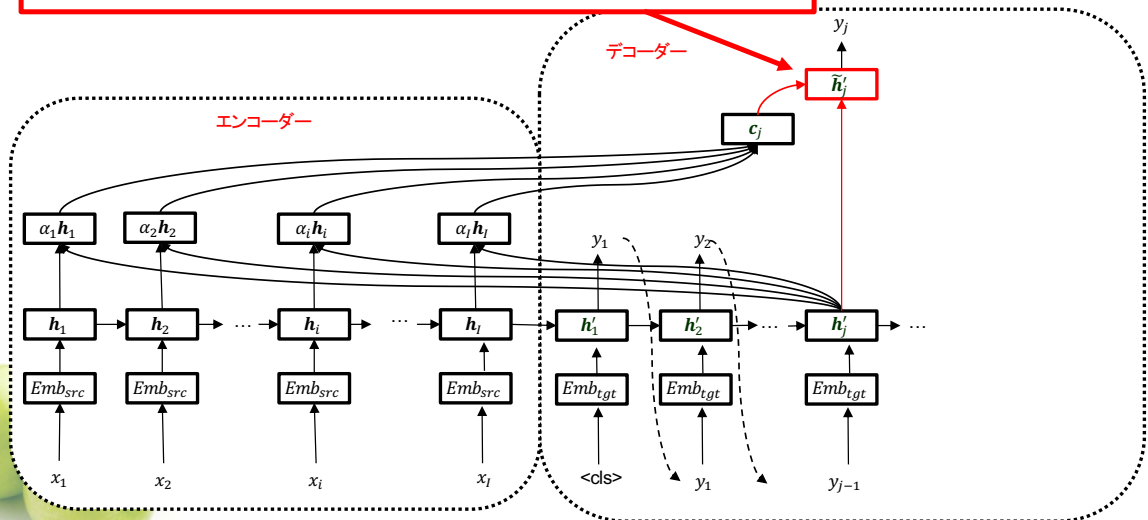
コンテキストベクトル c_j は α_i を重みとした h_i の加重平均となっています。

アテンション付き エンコーダー・デコーダーモデル

④ 新しい内部状態 \tilde{h}'_j を計算

$$\tilde{h}'_j = \tanh(W_c[c_j; h'_j] + b_c)$$

$[c_j; h'_j]$ は c_j と h'_j を連結したベクトル



新しい内部状態 \tilde{h}'_j を計算します。

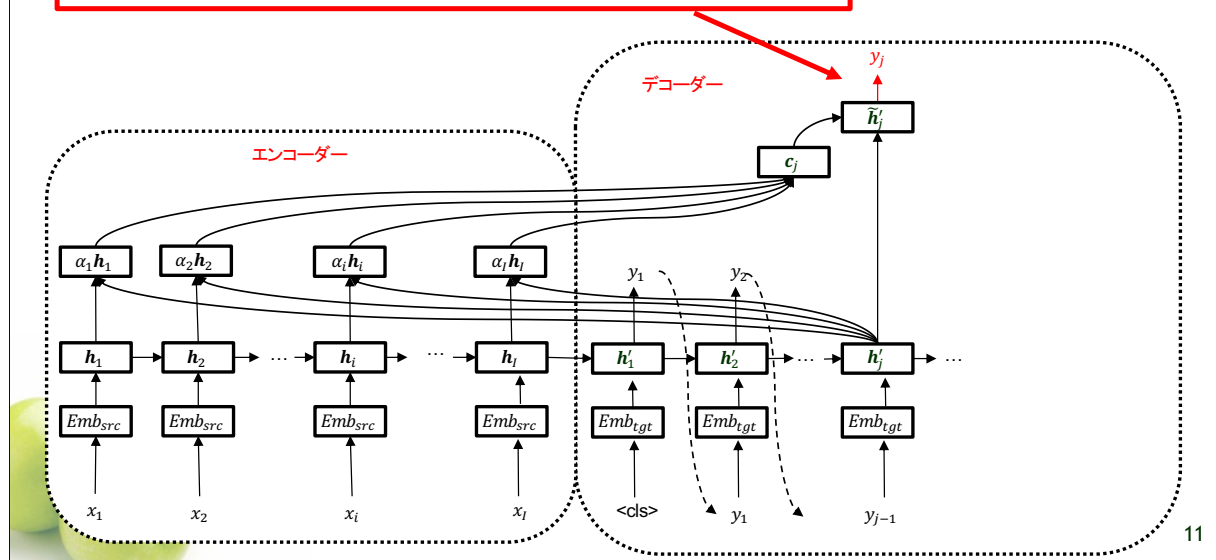
$$\tilde{h}'_j = \tanh(W_c[c_j; h'_j] + b_c)$$

$[c_j; h'_j]$ は c_j と h'_j を連結したベクトルとなっています。連結したベクトルに線形変換とハイパボリックタンジェントをかけることで新しい状態 \tilde{h}'_j を得ます。

アテンション付き エンコーダー・デコーダーモデル

⑤ 出力を計算

$$y_j = \text{softmax}(W_o \tilde{h}'_j + b_o)$$



最後に新しい状態 \tilde{h}'_j に対して線形変換とsoftmaxをかけて、単語出力の確率分布を得ます。

アテンション付き エンコーダー・デコーダーモデル

- **まとめ**

- エンコーダーの内部状態に対する重み付け

$$\alpha_i = \frac{\exp(\mathbf{h}_i^T \mathbf{h}'_j)}{\sum_{i'=1}^I \exp(\mathbf{h}_{i'}^T \mathbf{h}'_j)}$$

- 基本的に内積で計算されていることに注意
- $0 \leq \alpha_i \leq 1, \sum_{i=1}^I \alpha_i = 1$ となることに注意
- コンテキストベクトルは、エンコーダーの内部状態の加重平均

$$\mathbf{c}_j = \sum_{i=1}^I \alpha_i \mathbf{h}_i$$

- 新しい内部状態はコンテキストベクトルと内部状態を連結したベクトルに対する線形変換で得られる

$$\tilde{\mathbf{h}}'_j = \tanh(W_c[\mathbf{c}_j; \mathbf{h}'_j] + \mathbf{b}_c)$$



12

アテンション付きエンコーダー・デコーダーのまとめです。

アテンション付き エンコーダー・デコーダーモデル

- 重み付けのバリエーション

$$\alpha_i = \frac{\exp(\text{score}(\mathbf{h}_i, \mathbf{h}'_j))}{\sum_{i'=1}^I \exp(\text{score}(\mathbf{h}_{i'}, \mathbf{h}'_j))}$$

- Global Attention (dot)

$$\text{score}(\mathbf{h}_i, \mathbf{h}'_j) = \mathbf{h}_i^T \mathbf{h}'_j$$

ベクトル間の関係を内積で表現

- 先程説明したものと同一通常のアテンション

- Global Attention (general)

$$\text{score}(\mathbf{h}_i, \mathbf{h}'_j) = \mathbf{h}_i^T \mathbf{W}_a \mathbf{h}'_j$$

ベクトル間の関係を重み行列 \mathbf{W}_a で学習

- Global Attention (concat)

$$\text{score}(\mathbf{h}_i, \mathbf{h}'_j) = \mathbf{v}^T \tanh(\mathbf{W}_a [\mathbf{h}_i; \mathbf{h}'_j])$$

2つのベクトル間の重み付けを重み行列 \mathbf{W}_a と重みベクトル \mathbf{v} で学習



13

アテンションの重み付けについてはいくつかのバリエーションがあります。

Global Attention (dot)は先程説明したものと同一アテンションです。

$$\alpha_i = \frac{\exp(\text{score}(\mathbf{h}_i, \mathbf{h}'_j))}{\sum_{i'=1}^I \exp(\text{score}(\mathbf{h}_{i'}, \mathbf{h}'_j))}$$

としたとき、 $\text{score}(\mathbf{h}_i, \mathbf{h}'_j) = \mathbf{h}_i^T \mathbf{h}'_j$ で計算されます。

Global Attention (general)は、ベクトル間の関係を重み行列 \mathbf{W}_a で学習タイプのアテンションで、

$$\text{score}(\mathbf{h}_i, \mathbf{h}'_j) = \mathbf{h}_i^T \mathbf{W}_a \mathbf{h}'_j$$

により計算されます。

Global Attention (concat)は、2つのベクトル間の重み付けを重み行列 \mathbf{W}_a と重みベクトル \mathbf{v} で学習するタイプのアテンションで、

$$\text{score}(\mathbf{h}_i, \mathbf{h}'_j) = \mathbf{v}^T \tanh(\mathbf{W}_a [\mathbf{h}_i; \mathbf{h}'_j])$$

により計算されます。

どれもそんなに性能に差がないため、一番単純なGlobal Attention (dot)がよく用いられています。

TRANSFORMER



Transformer

- **Transformerモデル**

- 現在のニューラル機械翻訳のベースラインモデル
- RNNでもCNNでもない新しいタイプのニューラル機械翻訳モデル
- アテンションの計算だけで翻訳を行うことが特徴
- キーテクノロジー
 - 自己アテンション (Self Attention)
 - 位置エンコーディング (Positional Encoding)
 - マルチヘッドアテンション (Multi-head Attention)



15

続いて、Transformerについて解説します。

Transformerは、RNNでもCNNでもない新しいタイプのニューラル機械翻訳モデルで、現在のニューラル機械翻訳のベースラインモデルとなっています。アテンションの計算だけで翻訳を行うことが大きな特徴となっています。

Transformerには次のキーテクノロジーが使われています。

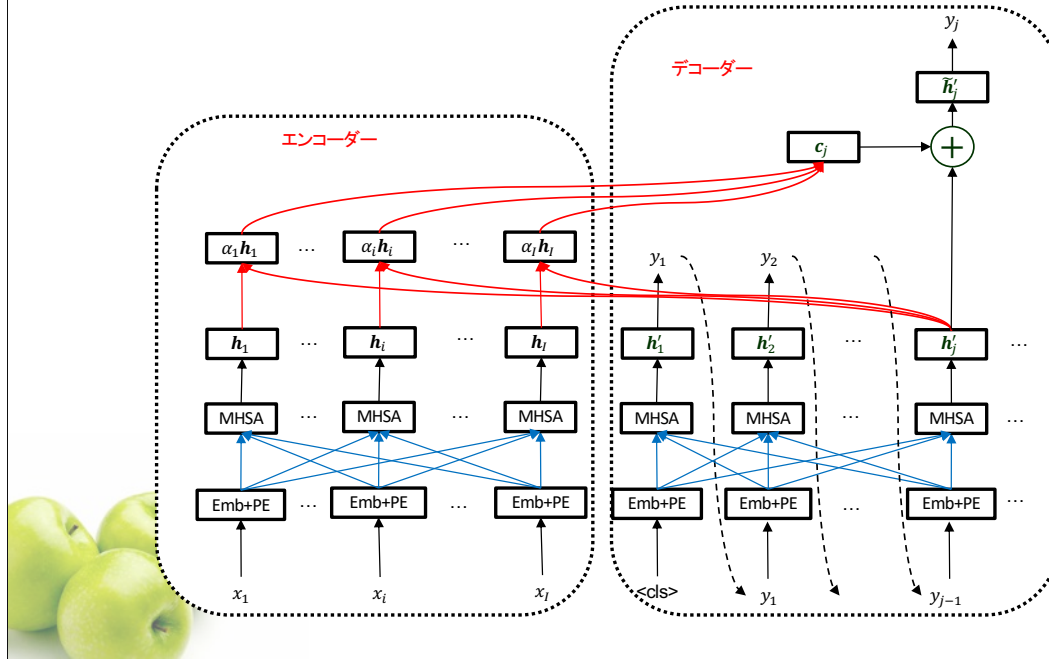
自己アテンション (Self Attention)

位置エンコーディング (Positional Encoding)

マルチヘッドアテンション (Multi-head Attention)

Transformer

- Transformerのモデル概要図



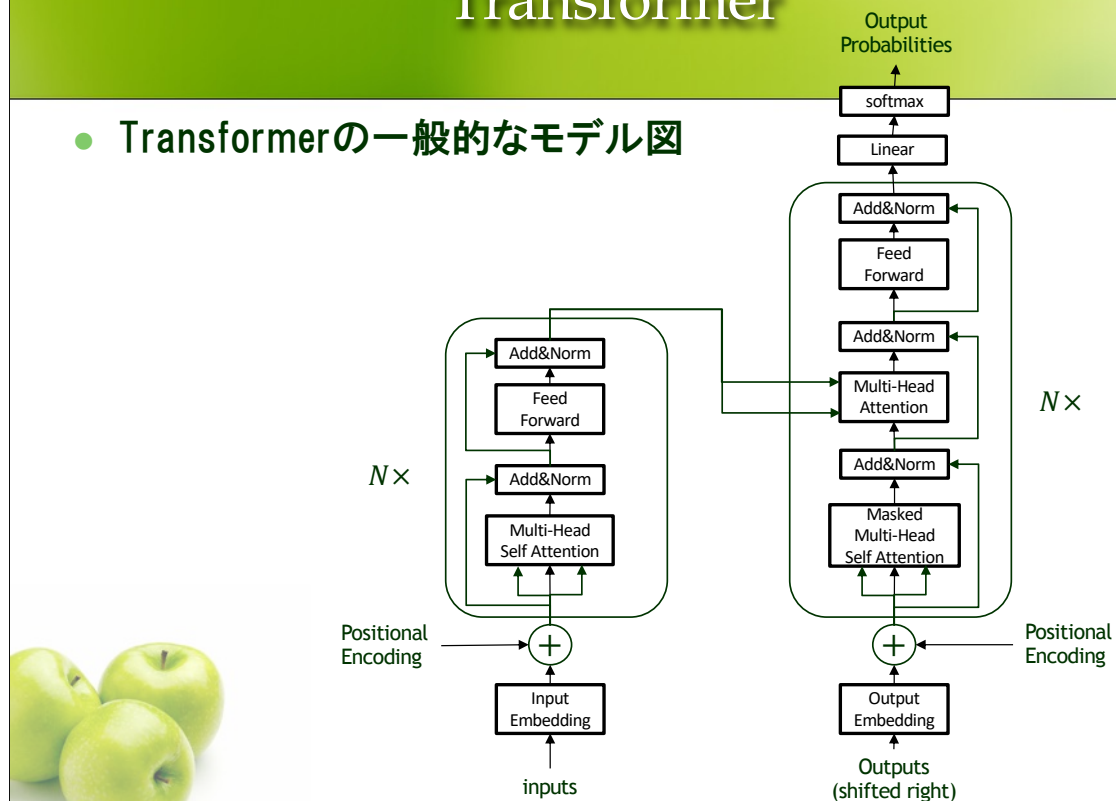
16

Transformerの全体図になります。アテンション付きエンコーダー・デコーダーモデルと同じような形をしています。大きな特徴としては、RNNのような状態遷移の計算がなく、青色の矢印で描かれた「自己アテンション」と赤色の矢印で描かれた「言語間アテンション」(アテンション付きエンコーダー・デコーダーモデルで説明した普通のアテンション)により主に計算されていることです。

自己アテンションでは、エンコーダー内の全ての単語間でアテンションの計算を行っています。デコーダーでも同じ様にデコーダー内の全ての単語間でアテンションの計算を行っています。言語間アテンションではデコーダーの注目している単語に対して、エンコーダー内の各単語に対するアテンションを計算していましたが、自己アテンションでは、同じ文内で、アテンションの計算を行っていることとなります。

Transformer

- Transformerの一般的なモデル図



一般にTransformerのモデル図はこのような図で表されています。単語列をまとめて処理するため、input=単語列として、まとめて表現します。

エンコーダーからまずみてみましょう。

入力はず単語埋め込み層によって、埋め込み表現に変換されます。その後、positional encodingによって、位置情報が単語埋め込みに付加されます。この計算は単純に単語埋め込みと位置エンコーディングの足し算を計算します。

続いて、マルチヘッド自己アテンションの層を通過し、Add&Norm層で再合流しています。Add&Normは残差接続(residual connection)とレイヤー正規化(layer normalization)を行っています。その後、feed forward層では単純な線形変換とReLUの計算を行っています。

次にデコーダー部をみてみましょう。エンコーダーと同じ様に、出力に対して、単語埋め込みと位置エンコーディングを行ったあと、(マスク付き)自己アテンションの計算を行い、Add&Norm層の処理を行います。

その後、エンコーダーから内部状態を受け取り、言語間アテンションの計算を行います。Add&Norm層の処理を行って、feed forward層の計算を行い、最後に線形変換とsoftmaxにより単語出力確率を計算します。

この図ではわかりにくいのですが、デコーダーは推論時には出力を再入力させる必要があることに注意してください。

Transformer

● アテンション

- d : 内部状態の次元数
- Query, Key, Valueの入力があると考えてアテンションを捉え直す
 - **Query(Q)**: 計算対象の内部状態 (デコーダーの内部状態)
 - **Key(K)**: アテンションの対象 (エンコーダーの内部状態)
 - **Value(V)**: アテンション先の内部状態 (エンコーダーの内部状態)
- **Q, K, V は(文長 $\times d$)の行列を表し、文全体のアテンションをまとめて計算**
- Q と K から重み(A)を計算して、重みと V の加重平均を計算。 A は(文長 \times 文長)の行列

$$A = \text{softmax}\left(\frac{QK^T}{\sqrt{d}}\right)$$

$$\text{Attention}(Q, K, V) = AV$$



18

Transformerではアテンションの計算を一般化して、Query, Key Valueの3種類の入力があると考えてアテンションを捉え直しています。

・Query(Q)は計算対象の内部状態で、今までのアテンションで言うと、デコーダーの内部状態にあたります。

・Key(K)はアテンションの対象で、今までのアテンションで言うと、エンコーダーの内部状態にあたります。

・Value(V)はアテンション先の内部状態で、今までのアテンションで言うと、エンコーダーの内部状態にあたります。

内部状態の次元数を d とすると、 Q, K, V は(文長 $\times d$)の行列を表していて、文全体のアテンションをまとめて計算しています。 Q と K から重み(A)を計算して、重みと V の加重平均を計算しています。 A は(文長 \times 文長)の行列で、

$$A = \text{softmax}\left(\frac{QK^T}{\sqrt{d}}\right)$$

によって計算されます。最終的なアテンションの結果(コンテキストベクトル)は、

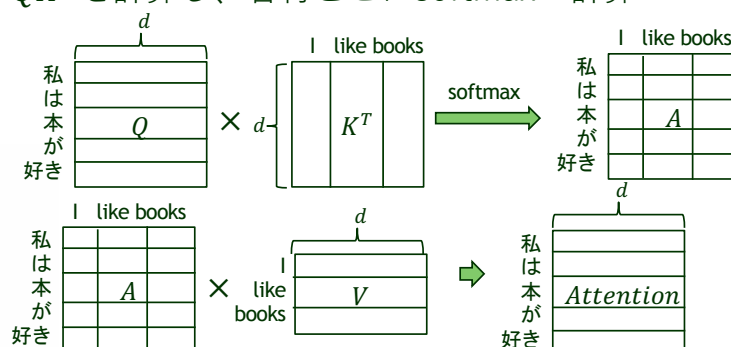
$$\text{Attention}(Q, K, V) = AV$$

によって計算されます。

Transformer: アテンションの例

- **言語間アテンション**(Encoder-Decoder Attention, Cross-Lingual Attention)

- エンコーダー(“I like books”)→デコーダー(“私 は 本 が 好き”)
 - Q : デコーダーの内部状態
 - K : エンコーダーの内部状態
 - V : エンコーダーの内部状態(K と V は同じ)
 - A : QK^T を計算し、各行ごとにsoftmaxの計算



19

一般化したアテンションを用いて、まず、言語間アテンション(encoder-decoder attention, cross-lingual attention)を定義します。言語間アテンションは今までアテンション付きエンコーダー・デコーダーモデルで定義してきたアテンションと同じものです。

エンコーダーに“I like books”が入力されて、デコーダーが“私 は 本 が 好き”を出力(入力)することを考えます。このとき、 Q はデコーダーの内部状態になり、 K, V はエンコーダーの内部状態になります。つまり、 K と V は同じになります。 A は QK^T を計算し、各行ごとにsoftmaxの計算を行って得られます。

Q はデコーダーの内部状態なので、“私 は 本 が 好き”の各単語に対する内部状態を表す行ベクトルを縦方向に並べた行列になります。 K はエンコーダーの内部状態なので、“I like books”の各単語に対する内部状態を表す行ベクトルを縦に並べた行列になるのですが、 K^T はこれを転置したものになるので、“I like books”の各単語に対する内部状態を表す列ベクトルを横に並べたものとなります。従って、 QK^T は“私 は 本 が 好き”と“I like books”の関連度を表す行列になります。

$$A = \text{softmax} \left(\frac{QK^T}{\sqrt{d}} \right)$$

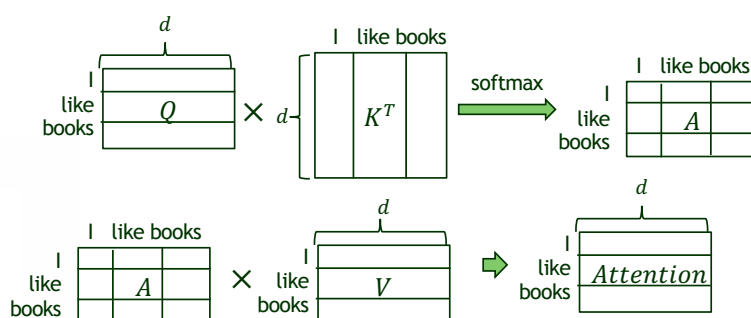
によって、 A の計算が行われるのですが、 QK^T を次元数のルートで割ったものに対し、各行毎にsoftmaxの計算を行います。

最後に AV を計算して、最終出力となるコンテキストベクトルを得ます。エンコーダーの文脈を考慮したデコーダー内部状態と考えてください。

Transformer: 自己アテンション

- 自己アテンション(Self Attention)

- エンコーダーの場合(“I like books”)
 - Q : エンコーダーの内部状態
 - K : エンコーダーの内部状態
 - V : エンコーダーの内部状態(Q と K と V は同じ)
 - A : QK^T を計算し、各行ごとにsoftmaxの計算



20

続いて新しいアテンションである自己アテンション(Self Attention)について説明します。エンコーダーとデコーダーの両方にそれぞれ自己アテンションがついているのですが、ここではエンコーダーを例に説明します。

エンコーダーに入力“I like books”が与えられているとします。自己アテンションでは、 Q, K, V の全てがエンコーダーの内部状態になります。つまり、自分自身の内部状態とアテンションの計算をしていることになります。

Q, K, V は全て“I like books”の各単語に対する内部状態を表す行ベクトルを縦方向に並べた行列になります。 K^T はこれを転置したものになるので、“I like books”の各単語に対する内部状態を表す列ベクトルを横方向に並べたものとなります。従って、 QK^T は“I like books”の文内の単語同士の関連性を計算していることとなります。

$$A = \text{softmax} \left(\frac{QK^T}{\sqrt{d}} \right)$$

によって、 A の計算が行われるのですが、 QK^T を次元数のルートで割ったものに対し、各行毎にsoftmaxの計算を行います。

最後に AV を計算して、最終出力となるコンテキストベクトルを得ます。文内の他の単語との関連性を考慮したエンコーダーの内部状態と考えてください。

Transformer

● マルチヘッドアテンション (Multi-head Attention)

- 内部状態を表すベクトル(\mathbf{h})をヘッド数(H)に分割して、それぞれの分割されたベクトルに対してアテンションの計算を行う
- d : 内部状態の次元数
- H : ヘッド数
 - 例: $d = 512, H = 8$ なら、各ヘッドの次元数は $512/8=64$ 次元)

- 線形変換による分割($W_h^Q, W_h^K, W_h^V \in \mathbb{R}^{d \times (d/H)}$)

$$Q_h = QW_h^Q, \quad K_h = KW_h^K, \quad V_h = VW_h^V$$

- 各ヘッドのアテンションの計算

$$\text{Head}_h = \text{softmax}\left(\frac{Q_h K_h^T}{\sqrt{d/H}}\right) V_h$$

- アテンションの計算($W^o \in \mathbb{R}^{d \times d}$)

$$\text{Attention}(Q, K, V) = [\text{Head}_1; \dots; \text{Head}_H] W^o$$



21

次にマルチヘッドアテンションについて解説します。アテンションは先程のようにも計算できるのですが、Transformerではさらに工夫をこらして、マルチヘッドという考え方を導入してアテンションの計算を行っています。

内部状態を表すベクトルは d 次元ベクトルなのですが、この d 次元ベクトルを H 個の小さなベクトルに分割します。それぞれの小さな内部状態ベクトルに対してアテンション計算を行い、それぞれの小さなアテンション機構のことをヘッドと呼びます。つまり、ヘッド数を H としたとき、 d 次元の内部状態を H 個の (d/H) 次元のベクトルに分割し、それぞれの小さなベクトルに対してアテンションの計算を行う、ということです。ヘッドが複数あるので、マルチヘッドアテンションと呼ばれます。例えば、 $d = 512, H = 8$ なら、各ヘッドの次元数は $512/8 = 64$ 次元となります。

分割に関して、単純にベクトルを H 個に分割するのではなく、線形変換により H 個のベクトルに分割します。すなわち、元の入力を Q, K, C としたとき、ヘッド h の Q_h, K_h, V_h は次のようにして求めます。

$$Q_h = QW_h^Q, \quad K_h = KW_h^K, \quad V_h = VW_h^V$$

ただし、 $W_h^Q, W_h^K, W_h^V \in \mathbb{R}^{d \times (d/H)}$ となります。

各ヘッドのアテンションの計算は先程までのアテンションの計算と同じで下記のようにして求められます。

$$\text{Head}_h = \text{softmax}\left(\frac{Q_h K_h^T}{\sqrt{d/H}}\right) V_h$$

最終的なアテンションの計算結果は、マルチヘッドでそれぞれ計算したアテンションの計算結果を連結して、連結されたベクトルに対して線形変換をして得られます。

$$\text{Attention}(Q, K, V) = [\text{Head}_1; \dots; \text{Head}_H] W^o$$

ただし、 $W^o \in \mathbb{R}^{d \times d}$ です。

Transformer

- **Feed Forward層**

- 各単語ごとの全結合層(Fully Connected Layer)2層+ReLU

$$FFN(x) = \max(0, xW_1 + b_1)W_2 + b_2$$



22

マルチヘッドアテンション層を抜けたあとは、Feed Forward層があります。これは単純に各単語毎の全結合層(Fully Connected Layer)2層+ReLUの層になります。

$FFN(x) = \max(0, xW_1 + b_1)W_2 + b_2$
によって計算されます。

Transformer

● レイヤー正規化 (Layer Normalization)

- ベクトル内の全ての次元の値の平均、分散を求めて正規化すること
- \mathbf{h} : 内部状態 ($\mathbf{h} = (h_1, \dots, h_d)^T$)
- d : 内部状態の次元数
- γ, β : パラメータ

$$\mu = \frac{1}{d} \sum_{i=1}^d h_i$$
$$\sigma = \sqrt{\frac{1}{d} \sum_{i=1}^d (h_i - \mu)^2}$$

$$\text{LayerNorm}(\mathbf{h}) = \gamma \odot \frac{\mathbf{h} - \mu}{\sigma} + \beta$$



23

続いて、マルチヘッドアテンション層とFFN層にそれぞれくっついているAdd&Norm層(残差接続とレイヤー正規化)について説明します。

まず、レイヤー正規化(Layer Normalization)から説明します。レイヤー正規化は、内部状態を表すベクトル内の全ての次元の値の平均、分散を計算して、それらによって正規化することです。

\mathbf{h} を内部状態 ($\mathbf{h} = (h_1, \dots, h_d)^T$)とし、 d を内部状態の次元数としたとき、このベクトルの各次元の値の平均 μ と分散 σ は次のようにして求められます。

$$\mu = \frac{1}{d} \sum_{i=1}^d h_i$$
$$\sigma = \sqrt{\frac{1}{d} \sum_{i=1}^d (h_i - \mu)^2}$$

このとき、レイヤー正規化層(LayerNorm)は、平均と分散を用いて、内部状態を正規化し、パラメータ γ, β を用いて線形変換(スケーリングとバイアス)することを行います。

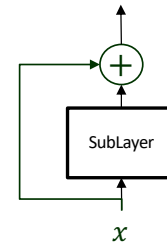
$$\text{LayerNorm}(\mathbf{h}) = \gamma \odot \frac{\mathbf{h} - \mu}{\sigma} + \beta$$

Transformer

- **残差接続(Residual Connection)**

- ニューラルネットワークのある部分ネットワークSubLayerがあったとき、次の計算を行うことを残差接続(Residual Connection)という

$$Res(x) = x + SubLayer(x)$$



- **Add&Norm層**

$$AddNorm(x) = LayerNorm(x + SubLayer(x))$$



24

Add&Norm層のもう一つの仕組みが残差接続(Residual Connection)です。残差接続は勾配消失を防ぐための一般的な手法で、深層化に必須の重要な技術となっています。(c.f. ResNet)

一般になんらかの部分ニューラルネットワークSubLayerがあったとき、次の計算を行うことを残差接続といいます。

$$Res(x) = x + SubLayer(x)$$

つまり、図のような計算を行うことになります。このように入力をそのまま出力にまわした接続と、SubLayerを通した接続を加算によってつなげている、と言えます。

Transformerのモデル図でのAdd&Norm層は、この残差接続とレイヤー正規化を行う層になっており、まとめて、

$$AddNorm(x) = LayerNorm(x + SubLayer(x))$$

の計算を行います。

Transformer: 位置エンコーディング

- 位置エンコーディング (Positional Encoding)

- 自己アテンションで同じ文内の任意の単語間の関係が計算されているが、単語間の距離が考慮されていない(隣の単語も遠くの単語も同じ様にアテンションの計算が行われる)
- 単語埋め込みの後に、位置情報を埋め込むことでこの問題を解決する

- 位置エンコーディングの式

- pos : 文中の位置
- $2i, 2i + 1$: 位置埋め込みベクトルの次元
- d : 埋め込みベクトルの次元数

$$PE_{(pos, 2i)} = \sin\left(\frac{pos}{10000^{2i/d}}\right)$$
$$PE_{(pos, 2i+1)} = \cos\left(\frac{pos}{10000^{2i/d}}\right)$$



25

ここまででTransformerの説明はほとんど終わりました。しかし、このままでは単語間の近さ、遠さに関する計算ができていない、という問題が残っています。自己アテンションで同じ文内の任意の単語間の関係が計算されていますが、単語間の距離が考慮されていません。つまり、隣の単語も遠くの単語も同じ様にアテンションの計算が行われてしまい、語順の違いなどがエンコーダー、デコーダーの計算に反映されません。

Transformerでは、単語埋め込みの後に、位置情報を埋め込むことでこの問題を解決します。次の式で求められる位置を表現したベクトルを単語埋め込みに加算して、位置情報付きの単語埋め込みにします。

位置エンコーディングの式

pos : 文中の位置

$2i, 2i + 1$: 位置埋め込みベクトルの次元

d : 埋め込みベクトルの次元数

$$PE_{(pos, 2i)} = \sin\left(\frac{pos}{10000^{2i/d}}\right)$$
$$PE_{(pos, 2i+1)} = \cos\left(\frac{pos}{10000^{2i/d}}\right)$$

Transformer

Dropoutとラベルスムージング

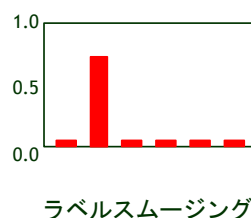
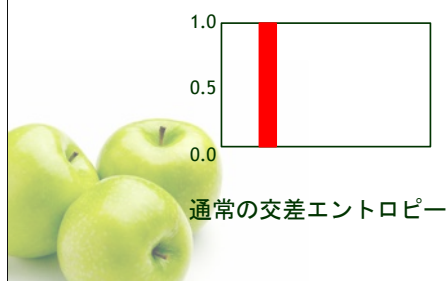
- 各層の後にDropoutを適用
- 出力にラベルスムージング(Label Smoothing)を適用

- 通常の交差エントロピー損失

$$L = - \sum_{j=1}^J \sum_{k=1}^K t_{jk} \log P(y_{jk} | x, y_{<j})$$

- ラベルスムージング

$$L = - \sum_{j=1}^J \sum_{k=1}^K \left[(1 - \epsilon) t_{jk} + \epsilon \frac{1}{K} \right] \log P(y_{jk} | x, y_{<j})$$



$t_{jk} \in \{0,1\}$ はj番目の出力の
k番目のラベルの正解

ϵ はスムージングの度合いを
調整するハイパーパラメータ

26

その他の細かな技術ですが、各層の後にDropoutを適用し、出力にラベルスムージング(Label Smoothing)という技術が用いられています。


ラベルスムージングは正解以外の出力にも僅かな確率を与える技術です。今までは正解の単語の出力確率が1で、その他は0としていましたが、正解の単語の出力確率を $1 - \epsilon$ として、それ以外の単語の確率を $\epsilon \frac{1}{K}$ とするスムージングの技術の一種となっています。交差エントロピー損失における正解ラベル $t_{jk} \in \{0, 1\}$ が $(1 - \epsilon)t_{jk} + \epsilon \frac{1}{K}$ となります。 ϵ はスムージングの度合いを調整するハイパーパラメータとなります。

Transformer: デコーダー

- マスク付きマルチヘッドアテンション

- デコーダーの自己アテンションの計算時にマスクをかける
 - 推論時には逐次的に単語が生成されるため、学習時にもマスクをかけて後ろの単語にアテンションがかからないようにする

デコーダーのアテンション行列A



| | 私 | は | 本 | が | 好き |
|----|---|---|---|---|----|
| 私 | | | | | |
| は | | | | | |
| 本 | | | | | |
| が | | | | | |
| 好き | | | | | |

黒い箇所を自己アテンションの計算対象にしない
(アテンションスコアを0にする)

27

基本的にはTransformerの説明は以上で終わりになるのですが、あとひとつ、デコーダーにおけるマスク付きマルチヘッドアテンションについて説明します。

推論時にデコーダーは、左から右に向かって一語ずつ出力します。その際には自己アテンションを前方に対してしか計算できないため、学習時にも前方だけをアテンションの対象として計算することが望めます。そのように前方だけをアテンションの対象とすることをマスク付きアテンションと呼びます。デコーダーの自己アテンションの計算時にアテンションをかけたくない後ろの単語に対して、マスクをかけます(アテンションスコアを0にする)。デコーダーのアテンション行列Aをみると、図の黒い部分がマスクがかかる部分になります。

ニューラル機械翻訳のまとめ

- アテンション付きエンコーダー・デコーダーモデル
- Transformer
 - 自己アテンション
 - マルチヘッドアテンション
 - レイヤー正規化
 - 残差接続
 - 位置エンコーディング
 - マスク付き自己アテンション



28

今回のまとめです。