

# 【セッション4】

## DaSiC2023 WORKSHOP

言語データとその「鏡」：  
機械学習モデルを用いた  
言い誤りと失語症例の  
分析

言語学とデータサイエンスに関する  
ワークショップ・  
参加無料

日時

**2023年**  
**12月23日〔土〕**  
**13:30 ~ 17:00**

モデルのデモンストレーション  
①百人一首 ②語彙の産出

全体討論

浅川伸一 (東京女子大学)  
吉原将大 (東北大学)

**December 23, 2023**



# Colab 操作方法

(デモンストレーション補足資料)

## **DaSiC2023 WORKSHOP**

資料作成：吉原将大

# 必要なもの

- パソコン or スマートフォン



- インターネット環境
  - Webブラウザは Chrome 推奨
  - Firefox, Safari も可



- Google アカウント
  - Colab の使用にはログインが必要



※本資料内のPC画面例は、実際とは異なる可能性があります

# 用語の説明

- コード（プログラム）のまとまりを「セル」と呼びます

セル

## 自作 Transformer の入力

```
[ ] 1 from RAM import Transformer
    2 model = Transformer(src_vocab_size=len(idx2tkn),
    3                     tgt_vocab_size=len(idx2tkn),
    4                     model_dim=32,
    5                     num_heads=4,
    6                     num_layers=1,
    7                     max_seq_length=22,
    8                     dropout=0.,
    9                     ff_dim=32).to(device)
    10 model.eval();
```

## 乱数の種の設定

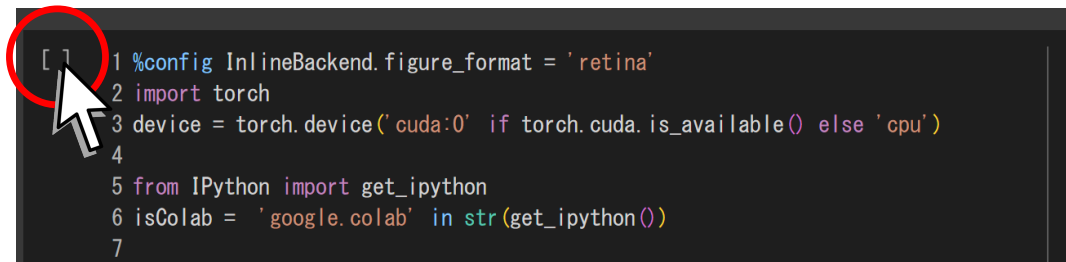
セル

```
[ ] 1 # 乱数のシードを設定
    2 import random
    3
    4 torch.manual_seed(42)
    5 np.random.seed(42)
    6 random.seed(42)
```

※コードに関する説明等

# Colab の基本的な使い方

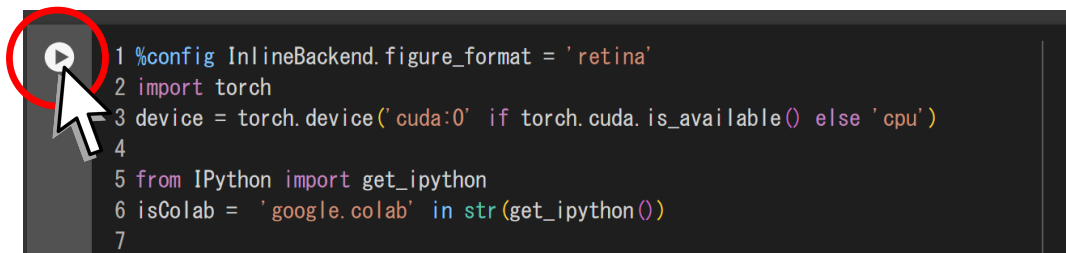
1. 各セルの左にある [ ] 上にカーソルを移動



A screenshot of a Google Colab code cell. On the left side of the cell, there is a small icon consisting of two brackets [ ] stacked vertically. A red circle is drawn around this icon, and a white mouse cursor arrow is pointing at it. To the right of the icon, there is a list of Python code lines numbered 1 through 7. The code is as follows:

```
1 %config InlineBackend.figure_format = 'retina'
2 import torch
3 device = torch.device('cuda:0' if torch.cuda.is_available() else 'cpu')
4
5 from IPython import get_ipython
6 isColab = 'google.colab' in str(get_ipython())
7
```

2. 再生マーク (▶) に変わったら左クリックしてコードを実行



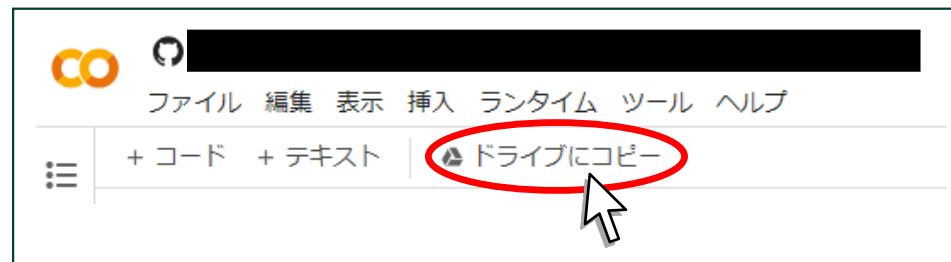
A screenshot of a Google Colab code cell, similar to the one above. The icon on the left now shows a play button (▶) inside the brackets [ ]. A red circle is drawn around this icon, and a white mouse cursor arrow is pointing at it. The code to the right is identical to the previous screenshot:

```
1 %config InlineBackend.figure_format = 'retina'
2 import torch
3 device = torch.device('cuda:0' if torch.cuda.is_available() else 'cpu')
4
5 from IPython import get_ipython
6 isColab = 'google.colab' in str(get_ipython())
7
```

※原則として、コードは上から順に実行してください

# デモのコードを自分の Google ドライブにコピー

- デモページを開いたら, 「ドライブにコピー」を左クリック
  - 新しいタブが自動で立ち上がり, ご自身の Google ドライブにデモンストレーションのコードが保存されます



- コピーしたコードであれば, ご自身でも修正・変更が可
  - 参考: Google ドライブの開き方 (右図)



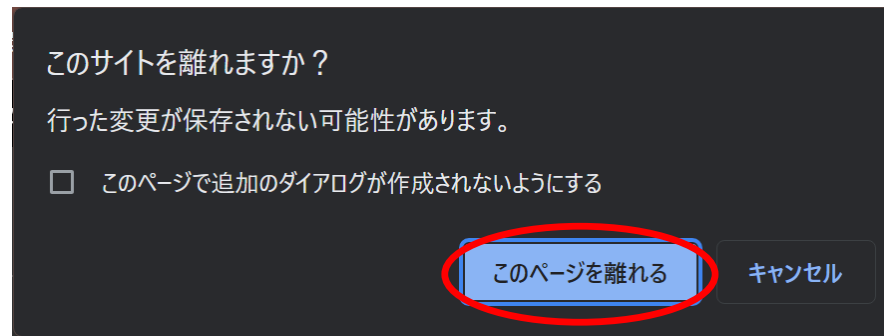
## もし警告が表示されたら

- セルの初回実行時に、以下のような警告が表示されることがあります
  - Google ドライブにコピーしたコードであれば、警告は表示されません
- （作成元が信用できる場合は）「このまま実行」を左クリック



# 終了時

- そのままブラウザを閉じてください
  - 以下の警告ウインドウは「このページを離れる」を左クリック





# 参考：Colab について



- 正式名称は “(Google) Colaboratory”
- Webブラウザ上で Python（プログラミング言語のひとつ）を記述・実行できる Google の機能
  - <https://colab.google/>
- 無料で GPU を使用可能
  - GPU (Graphics Processing Unit)：大量の計算を高速に実行できる
  - ただし、性能や使用量に上限がある（有料版だと上限が緩和される）

# 【セッション4】

## DaSiC2023 WORKSHOP

言語データとその「鏡」：  
機械学習モデルを用いた  
言い誤りと失語症例の  
分析

言語学とデータサイエンスに関する  
ワークショップ・  
参加無料

日時

**2023年**  
**12月23日〔土〕**  
**13:30 ~ 17:00**

モデルのデモンストレーション  
①百人一首 ②語彙の産出

全体討論

浅川伸一 (東京女子大学)  
吉原将大 (東北大学)

# SLAMモデル (Walker&Hickok2012) Semantic-Lexical-Auditory-Motor

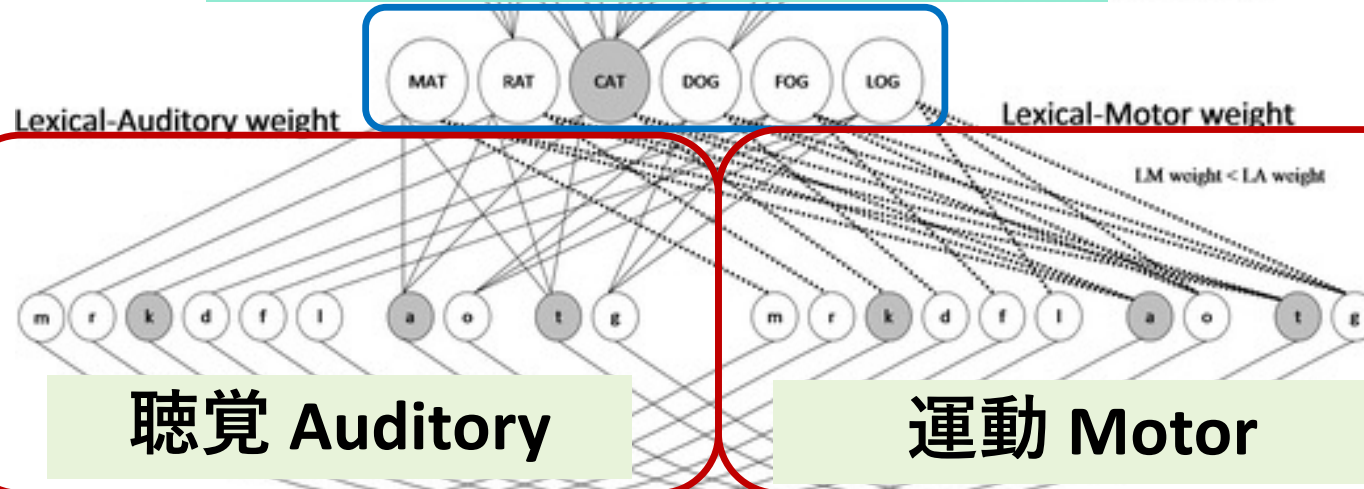
## Lexical-Auditory-Motor : LAM

語彙 Lexical  
NTT Data Baseより 1万語

語彙



音韻



聴覚 Auditory

運動 Motor

機械学習(符号化器-復号化器モデル)で実装  
注意機構を導入

健常モデル

試作モデル ～ LAMモデル Lexical–Auditory–Motor

## 事前学習 pretraining : 健常モデル

NTTデータベース 1K 語 [語彙L]  
音韻[聴覚A-運動M]の関係について  
機械学習(符号化器-復号化器モデル)で実装

別領域のデータを再学習させる手法

微調整 fine tuning : 全パラメタを固定しない

行動データ  
健常者の言い誤り および 失語症者の音韻性錯語  
“誤り再現率”をシュミレーション

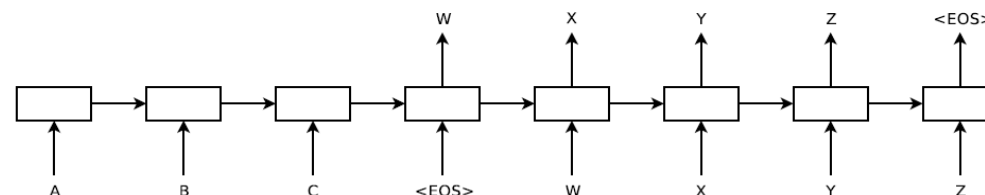
## 言語モデリングの分野で生まれた ([Sutskever, et al. 2014](#))

目的：入力系列 (ソース)  $\Rightarrow$  新しい配列 (ターゲット) に変換する

特徴：両系列は可変長の長さを持つことができる

例】テキストまたは音声の複数言語間の機械翻訳など

### seq2seq モデルの構成



**符号化器 encoder :**

入力系列を処理，情報を 固定長 の文脈ベクトルに圧縮する。

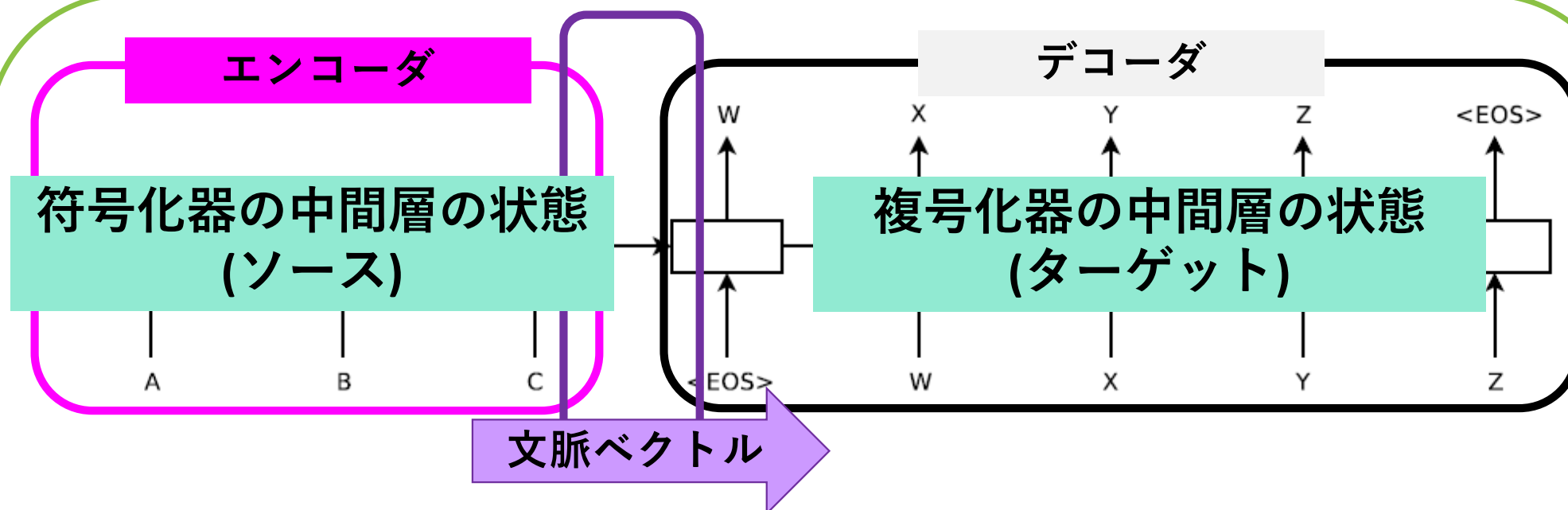
**復号化器 decoder :**

変換された出力を発するために，文脈ベクトルで初期化される。

## 符号化器-復号化器モデル

## 文脈ベクトル

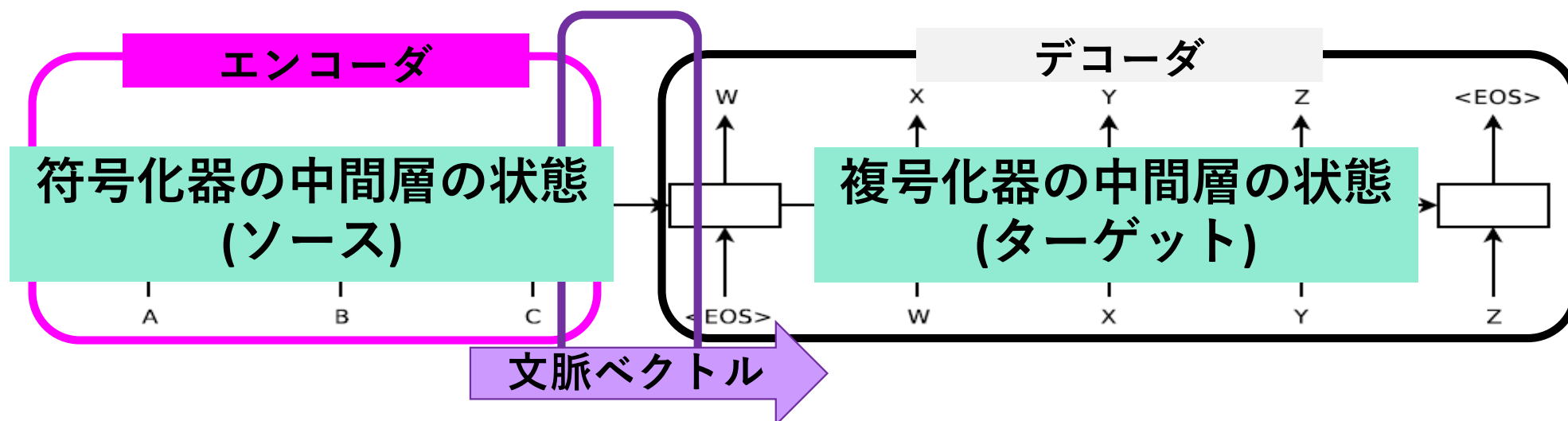
全体討論用



符号化器の最後の隠れ状態から単一の文脈ベクトルを構築するのではなく、  
文脈ベクトルとソース入力全体との間にショートカットを作成する。

ソースとターゲットの間のアライメントは、  
文脈ベクトルによって学習され、制御される。

中間層に加えた処理 ～ 注意機構 ～

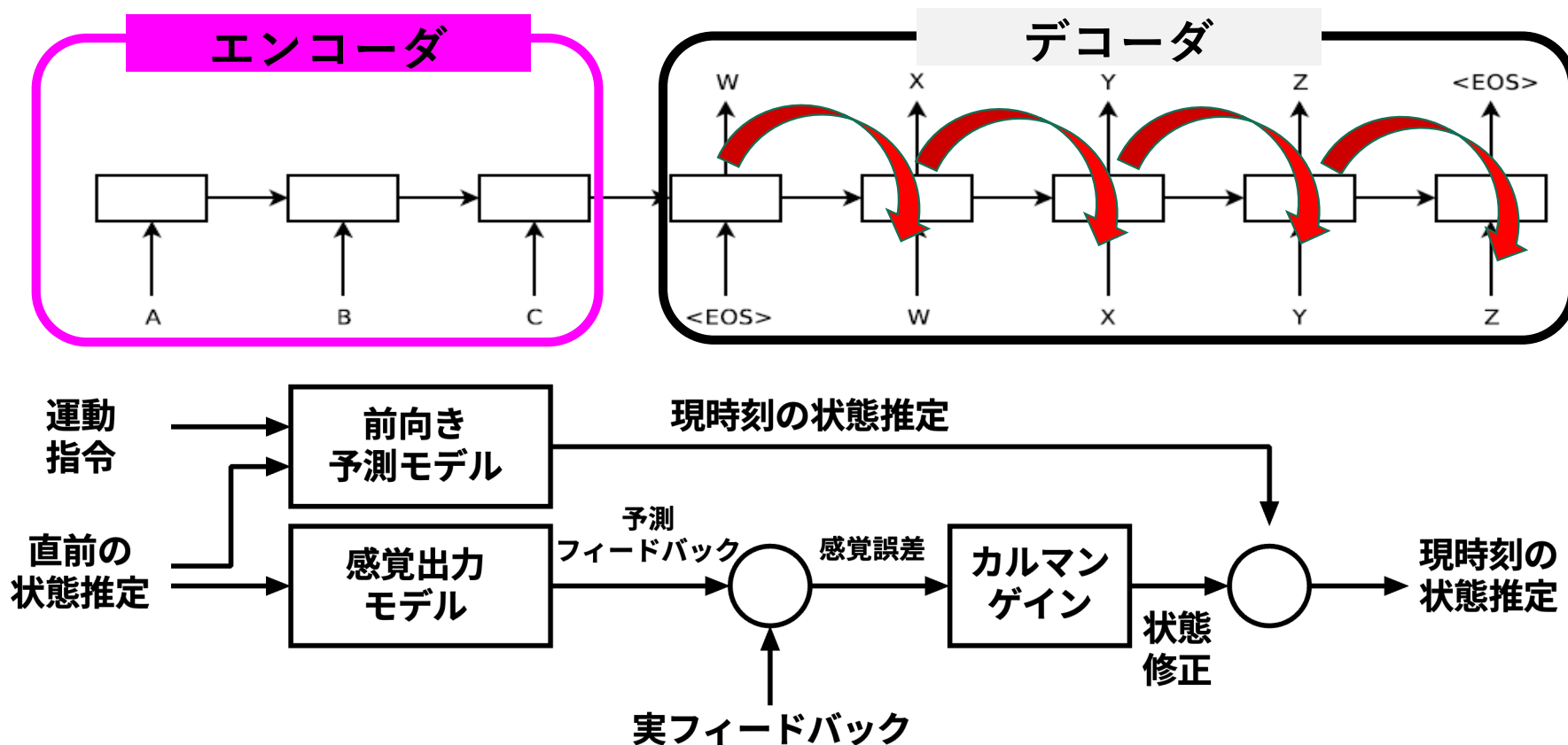


ソースとターゲットの  
アライメント(重み付け)

= “注意機構”

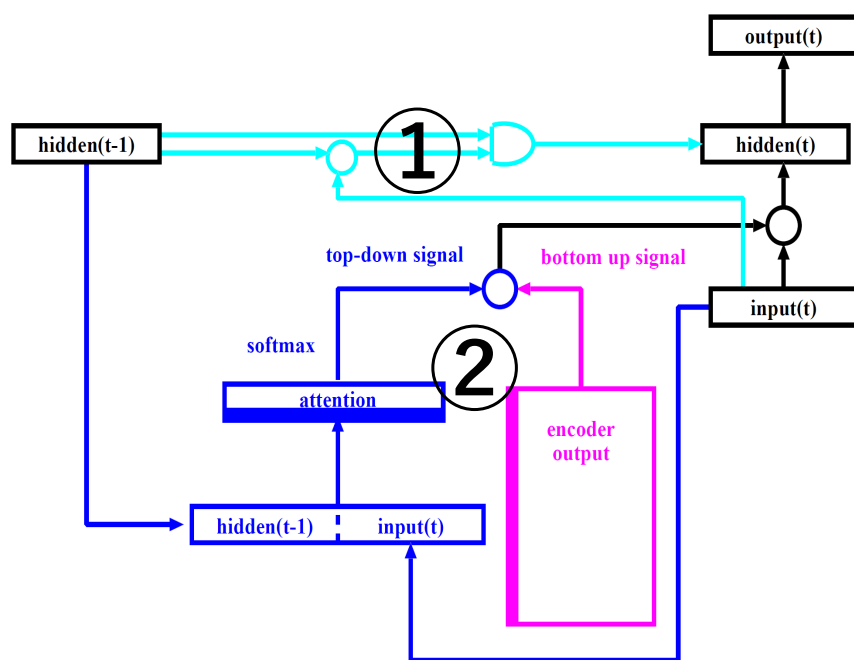
深層学習における注意とは、「重要度を表す重みベクトル」  
(勝者占有回路, winner-take-all circuit)

## 中間層に加えた処理 ～ カルマンフィルタ ～





### 中間層に加えた 3つの処理



### ①ゲート

#### 聴覚表象と運動表象のインターフェイス

(SLAMモデルにおけるAM部)

[仮定: 音素配列規則の逐次運動変換]

### ②注意機構

#### トップダウン注意

デコーダ(出力)からのフィードバック機構

[仮定: 運動情報からのフィードバック]

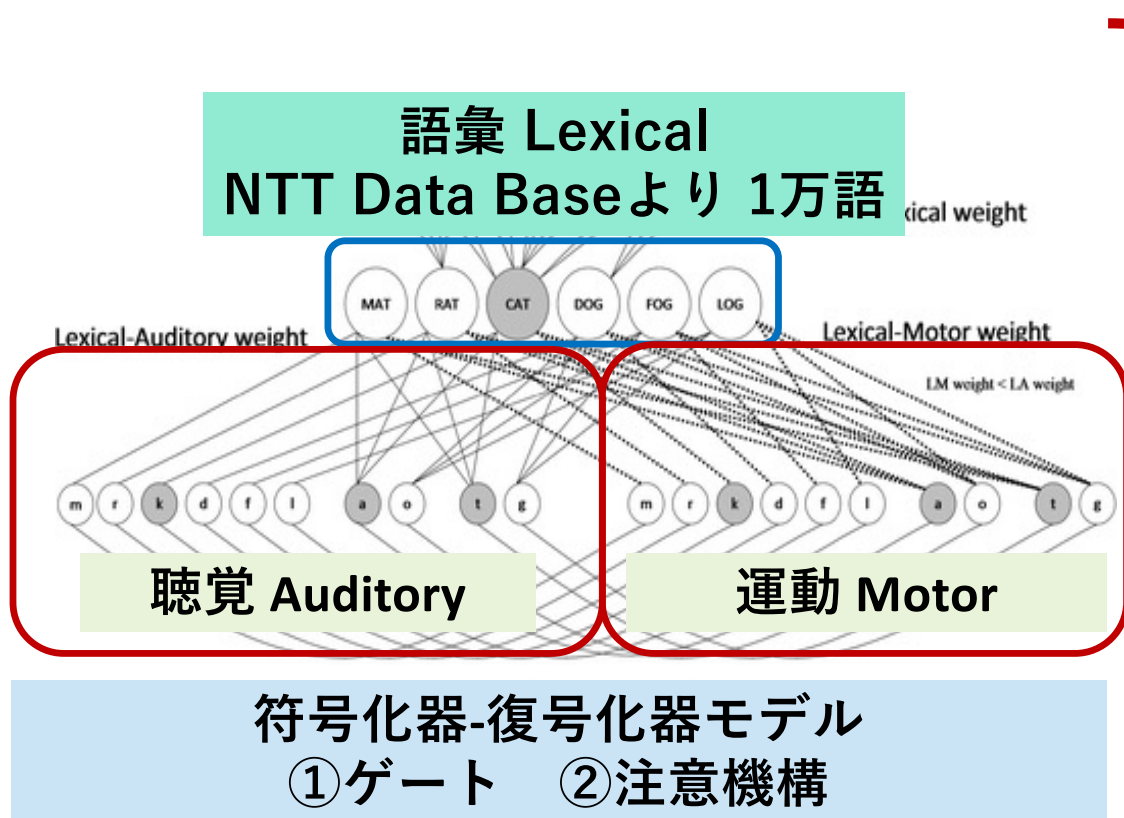
#### ボトムアップ注意

エンコーダ(入力)からのフィードバック機構

[仮定: 聴覚情報からのフィードバック]

おまけ  
音韻部門の復唱  
シミュレーション

# 考察 ～ LAMモデル Lexical-Auditory-Motor ～



健常モデル

健常者



いどろり

失語症者



だいとん

微調整  
fine tuning

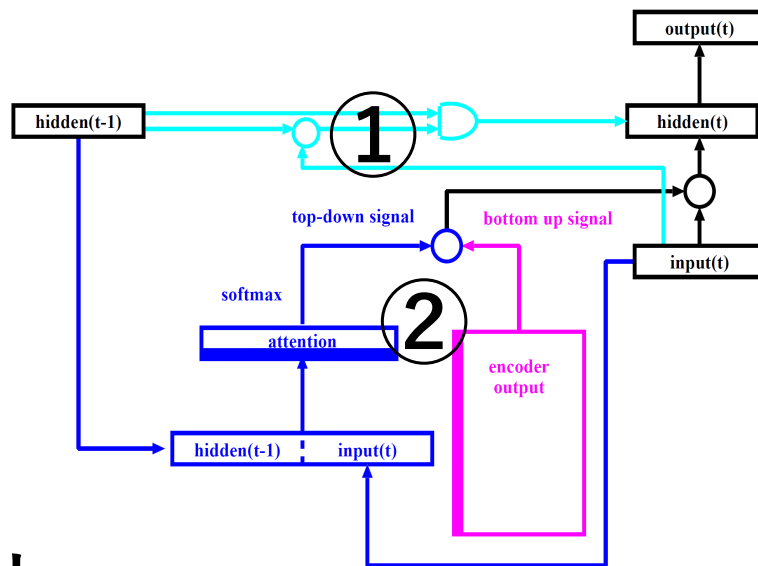


言い誤りの生成  
成功



## 結果 ～健常者の言い誤り～

### 健常モデルに対して音韻的交換型の言い誤りにて、微調整



#### ①ゲート

聴覚表象と運動表象のインターフェイス

[仮定: 音素配列規則の逐次運動変換]

#### ②注意機構

聴覚情報と運動情報からのフィードバック

モデル0 ①+②を微調整

言い誤り再現数：143/144

言い誤り再現率：99.3%

モデル1 ①のみ微調整

言い誤り再現数：139/144

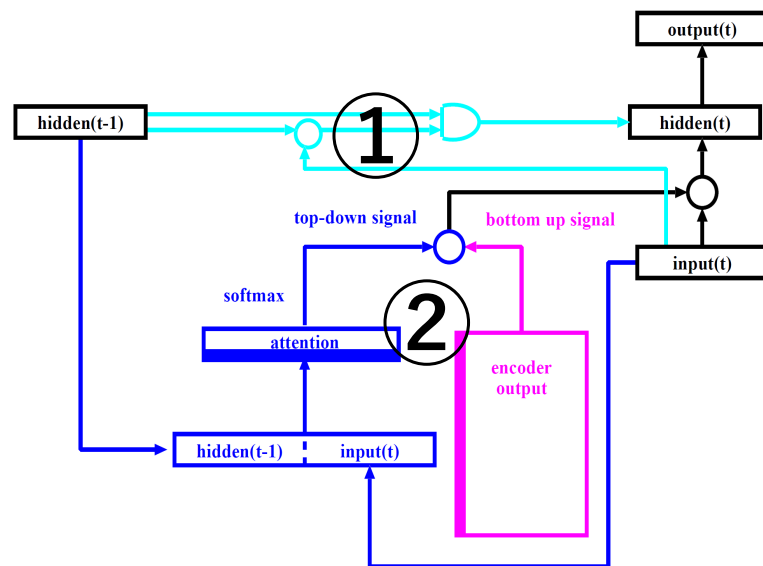
言い誤り再現率：96.5%

モデル2 ②のみ微調整

言い誤り再現数：52/144

言い誤り再現率：36.1%

## 結果 ～失語症者の言い誤り～ 健常モデルに対して音韻性錯語にて、微調整



※音韻性錯語データでは、1刺激語に対して、複数回答があった場合も計上した為に再現率が低下した。  
1 刺激に対して1つの言い誤りデータであれば、  
モデル0は100%となる。

モデル0 ①+②を微調整  
言い誤り再現数：50/65  
言い誤り再現率：76.9%

モデル1 ①のみ微調整  
言い誤り再現数：45/65  
言い誤り再現率：69.2%

モデル2 ②のみ微調整  
言い誤り再現数：36/65  
言い誤り再現率：55.3%

## 結果 ～ まとめ ～

### ①ゲート

聴覚表象と運動表象のインターフェイス  
[仮定: 音素配列規則の逐次運動変換]

### ②注意機構

聴覚情報と運動情報からのフィードバック

健常者でも失語症者でも：  
モデル0で再現率が高い

健常者の言い誤り：  
モデル1が高く、モデル2が低い

失語症者の音韻性錯語：  
モデル1もモデル2も低い

