SB Intuitions

2024.6.25　LLM勉強会

Sarashina: SB Intuitionsの日本語事前学習モデルの紹介

SB Intuitions株式会社

高瀬 翔 ｜ Sho TAKASE

自己紹介

- 2008-2017：東北大学（学士-博士）
- 2017-2018：NTT CS研（ポスドク）
- 2018-2022：東工大（研究員 → 助教）
- 2022-　　：LINE → LINEヤフー株式会社
- 2023-　　：SB Intuitions（出向）

- 自然言語処理の研究に従事
  - 特に系列変換タスクに取り組む
    - 機械翻訳・要約生成など
    - 2017年からニューラル言語モデルの研究もしている（IJCNLP 17, EMNLP 18, AAAI 19など）
  - 最近は効率的なニューラルモデルの研究に従事
    - 効率に関する研究論文：NeurIPS 20, NAACL 21, ACL Findings 22, 23 などに採択

LINEでも大規模言語モデルをつくってました

- LINEが公開している日本語言語モデル構築に参加
- 事前学習：高瀬・清野舜さん
- インストラクションチューニング：小林滉河さん・水本智也さん

36億パラメータの日本語言語モデル
を公開しました

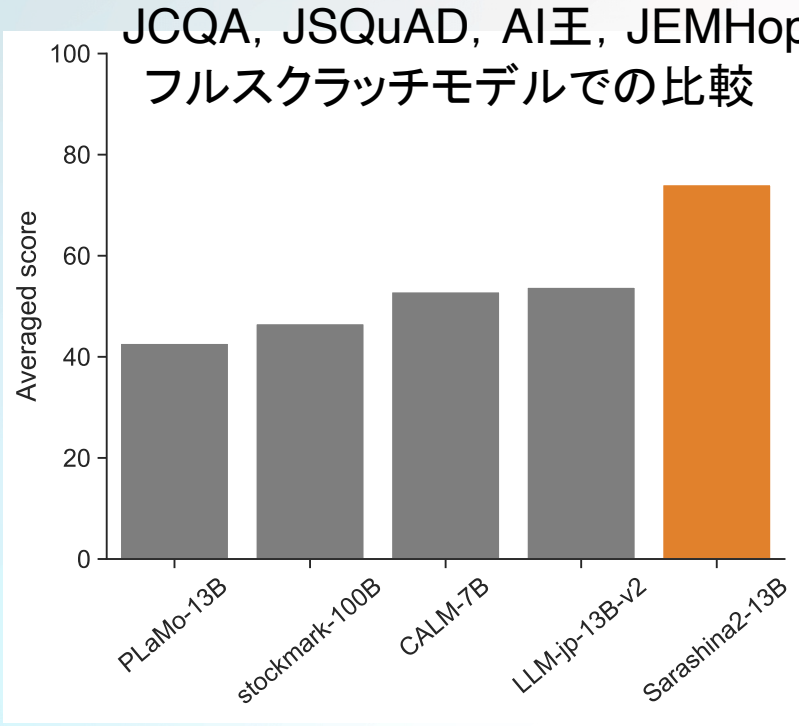
Shun Kiyono, Sho Takase, Toshinori Sato(overlast) 2023-08-14
NLP Foundation Dev Team

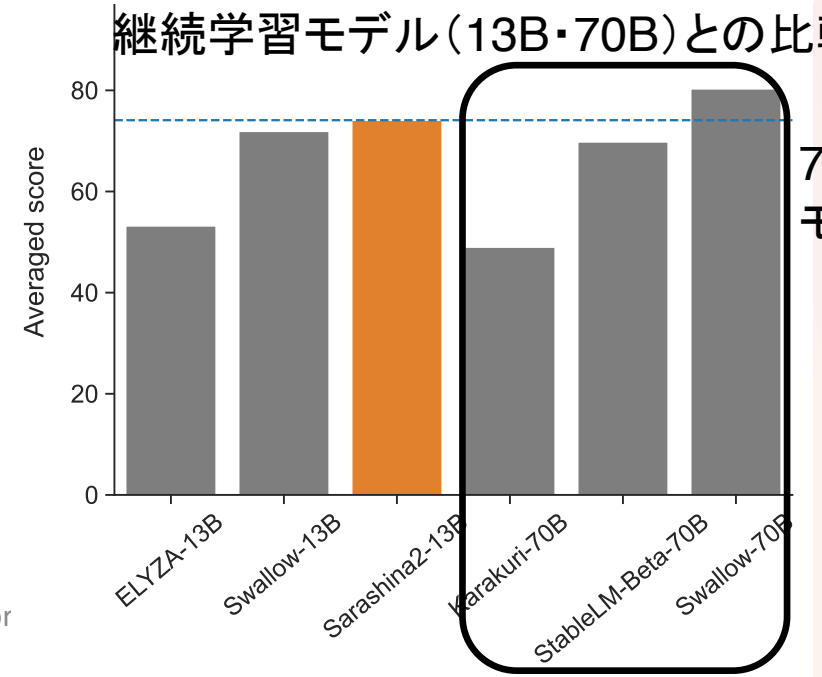
LINE Engineering ブログより

SB Intuitions から日本語言語モデルを公開

- 2024/06/24現在で公開されている日本語言語モデルと比較すると…
  - フルスクラッチで学習したモデルでは最も高い性能
  - 同程度のパラメータ数のモデルとの比較でも最も高い性能
    - Swallow-70B より低いだけという見方もある
- ただし事前学習モデルのみ

JCQA, JSQuAD, AI王, JEMHopQA, NIILC-QA の平均スコア（スコア計算は内製の評価コード）





公開した言語モデルの特徴・仕様

	Sarashina1	Sarashina2
訓練データ	日	日・英・コード
訓練トークン数	1T	2T
語彙数	51200	102400
学習率	2e-4 / 1e-4 / 6e-5	2.5e-4
Weight decay	0.01	0.1
Auxiliary z-loss	1e-4	1e-4
Scaled embed	◯	◯

公開した言語モデルの特徴・仕様

	Sarashina1	Sarashina2
訓練データ	日	日・英・コード
訓練トークン数	1T	2T
語彙数	51200	102400
学習率	2e-4 / 1e-4 / 6e-5	2.5e-4
Weight decay	0.01	0.1
Auxiliary z-loss	1e-4	1e-4
Scaled embed	◯	◯

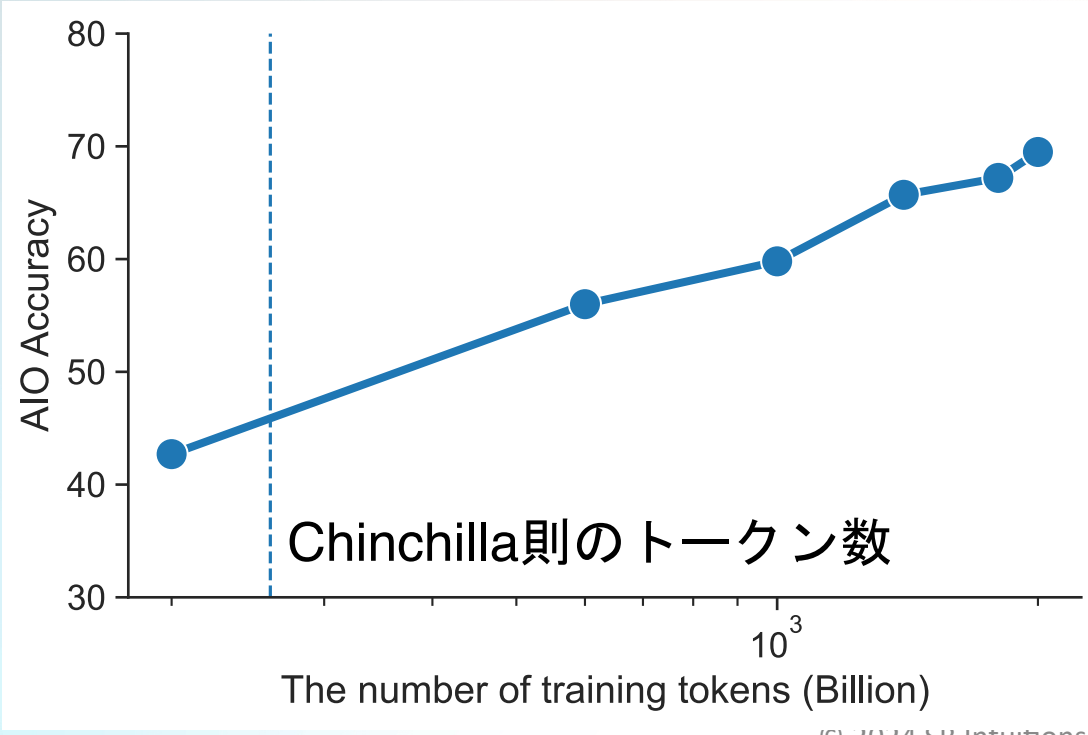
この項目を解説します

訓練データ

- 日本語：CommonCrawl から抽出
  - CCNet で言語判定と重複文書除去
  - Hojichar でルールベースや品詞情報を元にしたクリーニングなどを行う
- 英語：SlimPajama w/o (Books + Github) + Project Gutenberg
- コード：starcoderdata
- 比率は日:英:コードで5:4:1なのでコードはそれほど入ってない

訓練トークン数

- Llama2 を参考に 2Tトークンに設定
  - Chinchilla則（20トークン / パラメータ）よりもかなり大きい値の使用が流行
  - 性能は訓練トークン数に対数比例するので多いほど良い



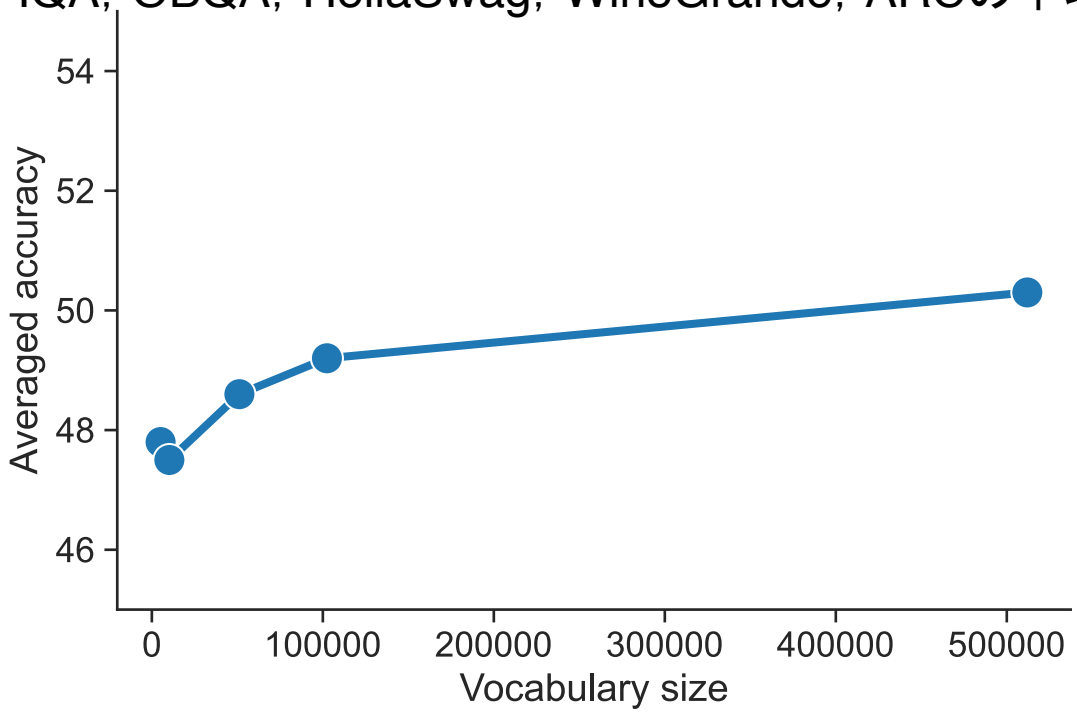
13Bパラメータでの訓練トークン数に対するAI王の正解率

Chinchilla則を超えても性能は上がる
→ Chinchilla則以上の学習が最近の流行（e.g., Llama シリーズ）

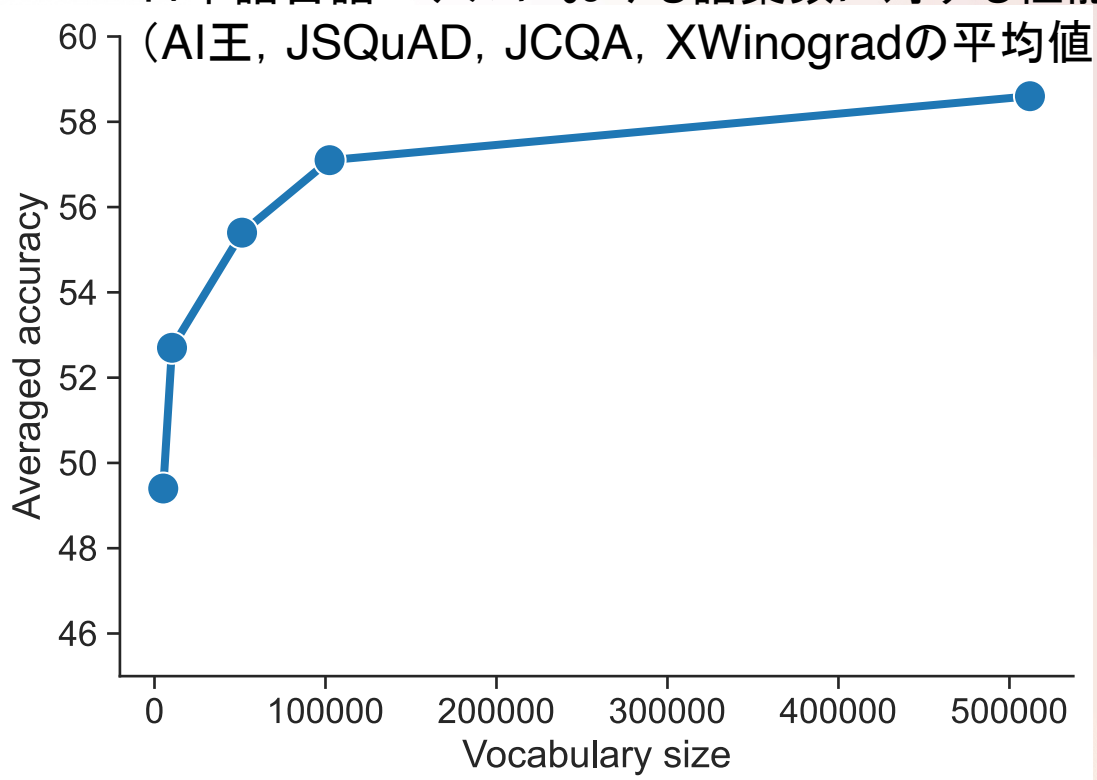
語彙数は大きいほど性能が良い

- 語彙数は大きいほど性能が良いので 50k → 100k に
  - 下図は中間層を680Mパラメータとしたときの語彙に対する性能
  - 近々 arXiv に論文を出す予定

英語言語モデルにおける語彙数に対する性能
（PIQA, OBQA, HellaSwag, WinoGrande, ARCの平均値）

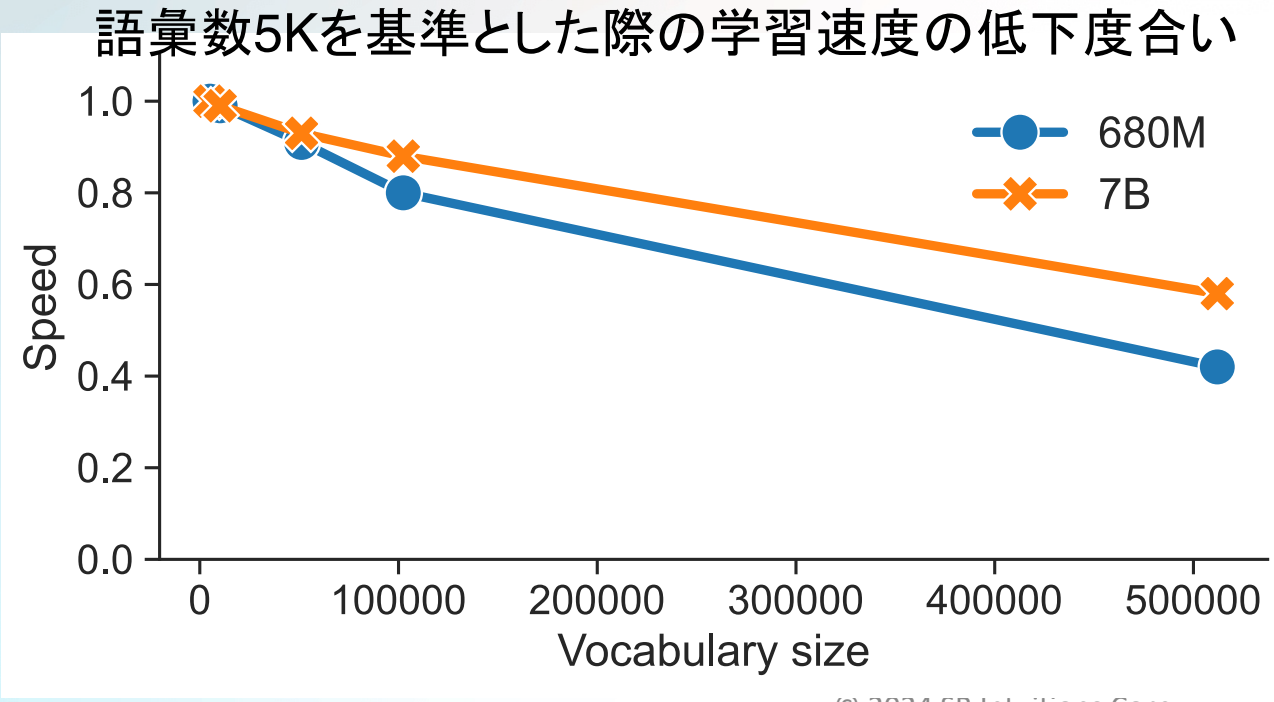


日本語言語モデルにおける語彙数に対する性能
（AI王, JSQuAD, JCQA, XWinogradの平均値）



大きすぎる語彙数は計算効率に悪影響

- 語彙数が大きくなると出力確率分布の計算が重くなる
  - 語彙数 500k の速度は 5k の半分程度
  - 中間層のパラメータ数が大きくなるほど影響は小さくなるが……

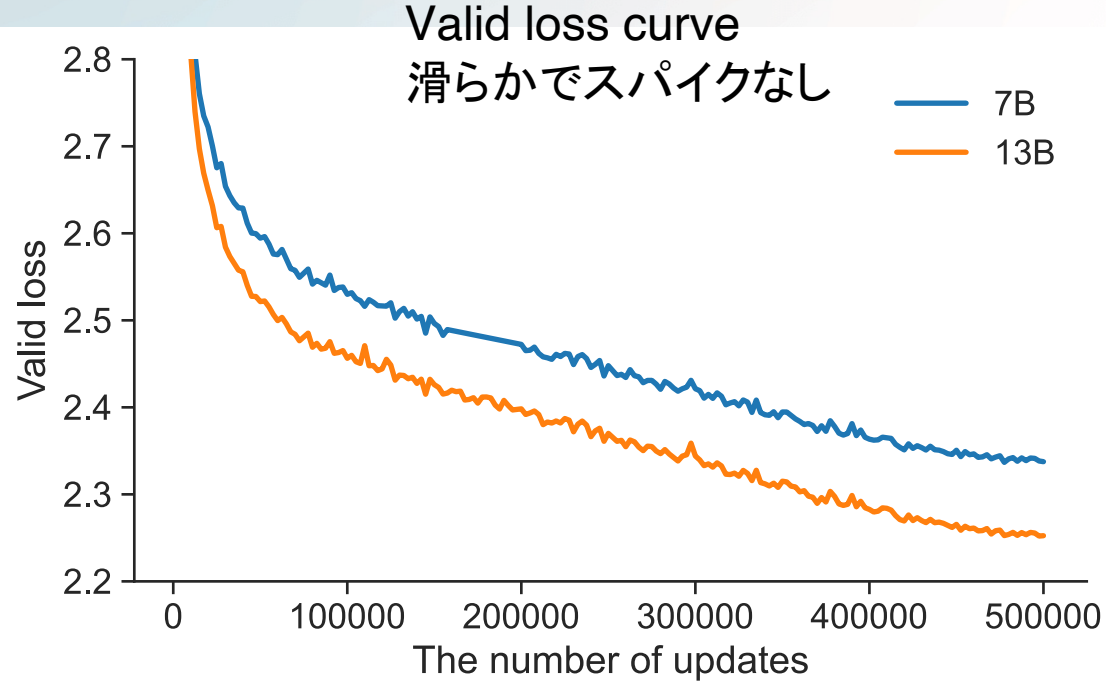


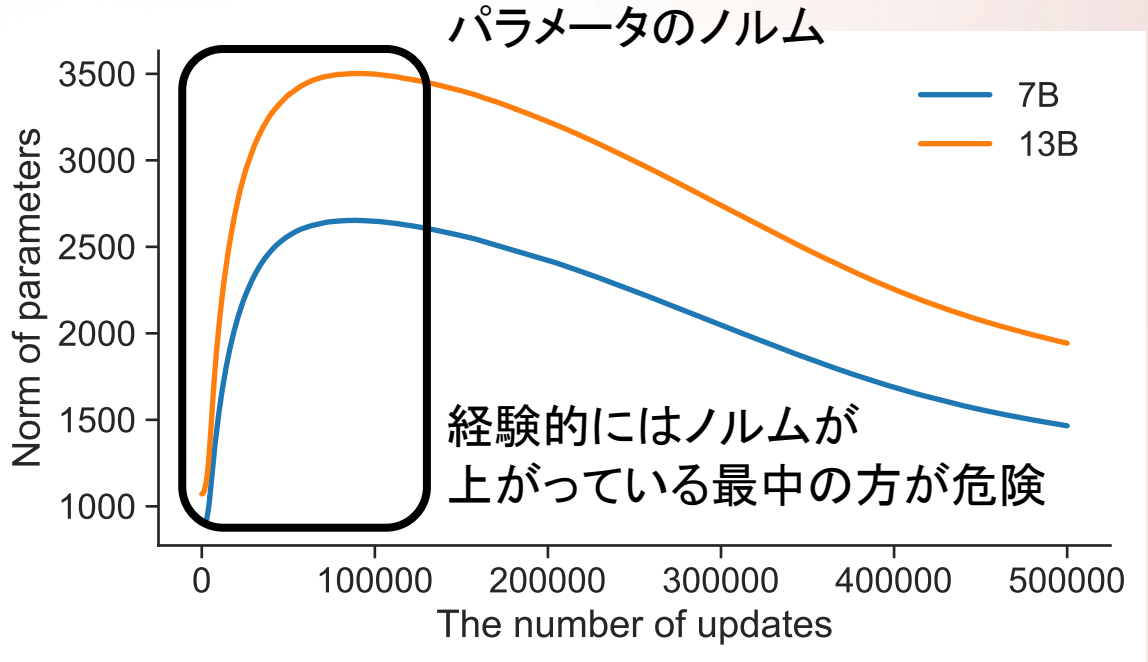
語彙数に比例して計算は遅くなる
許容できるのは 100k 程度に思える

語彙数をより大きくしたい場合は
単語を扱っていた時代のテクニックが
必要かもしれない（e.g., adaptive softmax）

学習率を Llama の値に近づける

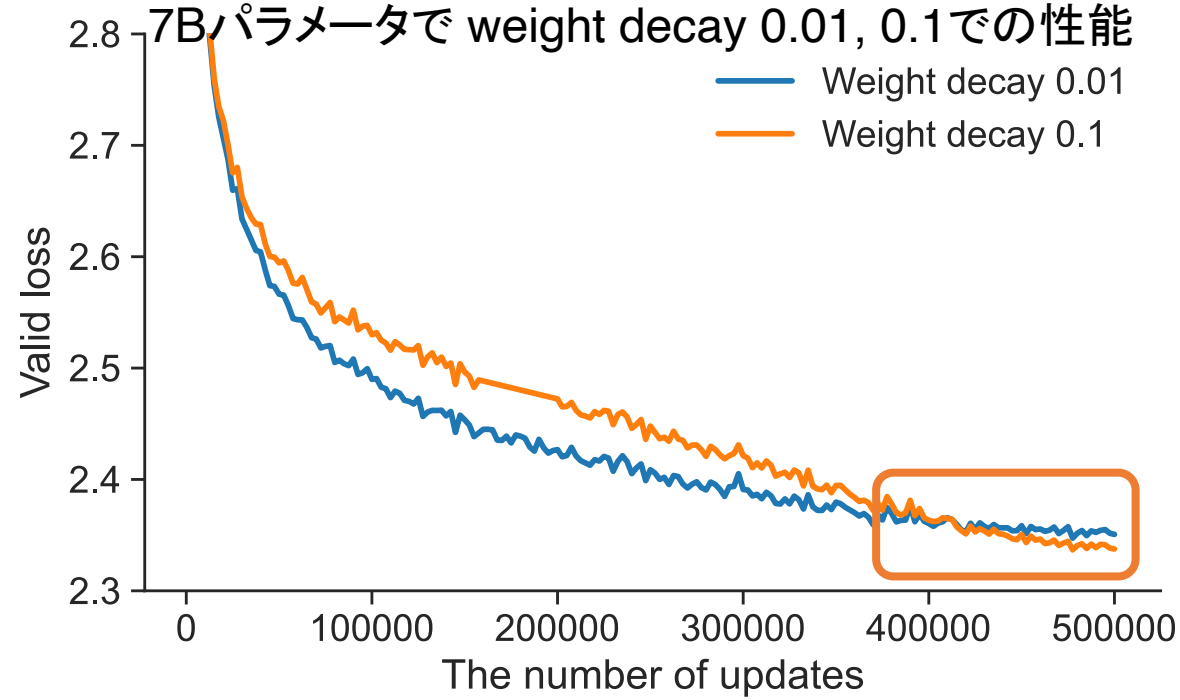
- 高いほど性能が良いが学習も失敗しやすくなる
  - Sarashina2 は 2.5e-4 を使用
  - Llamaの 7B・13B での 3.0e-4 を参考に





Weight decay を大きくする

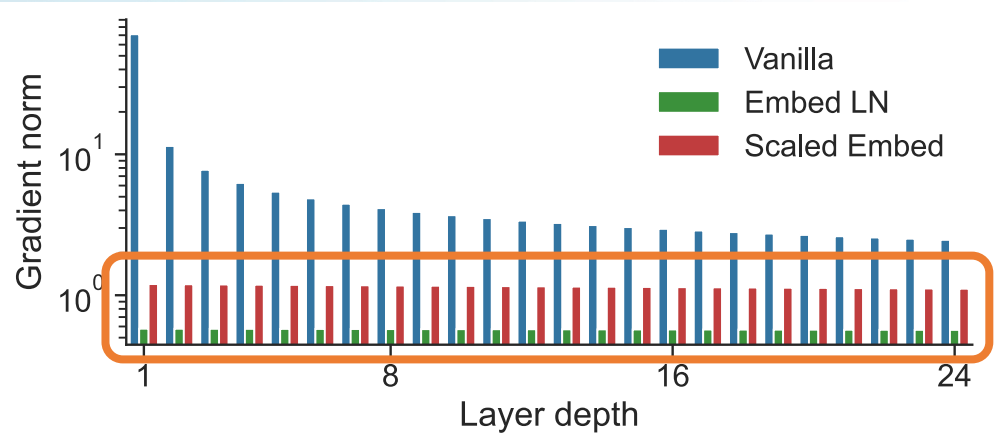
- Weight decay は0.1にしておけば良さそう
  - 0.01 の方が学習初期の性能は良いが後半で 0.1 の性能が上回る
  - 0.1 の方が学習の安定性も高い



学習後半で weight decay 0.1 の性能が 0.01 を上回る

Scaled embed を入れる

- 埋め込み表現の分散を 1 に近づけると各層の勾配が均一になる
  - LNでの勾配の増幅が抑制される
  - 学習の安定性に効果があるとされている
- 分散が 1 に近づけば何でも良い
  - 例：PaLM は平均0，分散1 の標準正規分布で初期化
  - 今回は埋め込み表現に √d をかけ合わせる（Scaled embed）



埋め込み表現の分散が 1 に近いと各層の勾配が均一になる

図はTakase et al., Spike No More: Stabilizing the Pre-training of Large Language Models から引用

まとめ

- SB Intuitions から公開した日本語事前学習モデルの紹介
  - フルスクラッチで学習したモデルでは最も高い性能
  - 同程度のパラメータ数のモデルとの比較でも最も高い性能

- 学習設定は既存研究の報告を整理して導入
  - Llama の学習率は高すぎる気がしていたが意外となんとかなるのかも…