

microgpt を解剖する（高校生・一般向け）

4つの名前だけで、AI が名前を作るしくみを一つずつたどる

Planned and edited by Yoshio Okamoto / Written by Claude Opus 5.5 (Anthropic)

Ver. 0.1 October 21, 2026

※ 以下の内容はAIが生成したものであり、一応編者がチェックしていますが、間違いや思い込みが含まれている場合があることはご注意ください。

© 2026 Yoshio Okamoto. 教育目的での利用は自由です（出典を明記してください）。microgpt のプログラムは Andrej Karpathy の著作物です。

はじめに

ChatGPT や Claude のような AI（大規模言語モデル，Large Language Model，LLM）は、とても単純なことをくり返しています。これまでの文章の続きとして、次に来る言葉を一つ予想し、それを付け足して、また次を予想する。これだけです。

microgpt は、Andrej Karpathy が 2026年2月に公開した、約200行の Python のプログラムです。大きな AI と同じしくみを、ごく小さな規模で動かします。扱うのは文章ではなく英語の名前で、「次の文字」を当ててることを学び、最後に新しい名前を作ります。

この冊子では、microgpt に **emma**, **ava**, **mia**, **anna** の4つの名前だけを学ばせて、そのはたらきを最初から最後まで解剖します。小さくすると、AI の中で動くすべての数を、紙の上で一つずつ追いかけるからです。本文の数は、すべて実際にプログラムを動かして確かめたものです。

章	内容	見るもの
1	全体像と準備	AI は何を練習するのか。データ、文字の番号、練習問題
2	1文字を読む	emma を1文字ずつ、実際の数で計算をたどる
3	外れ具合と学習	罰点（損失）を測り、つまみを少しずつ回す
4	名前を作る	確率でくじを引き、新しい名前を作る
5	本物の AI と比べる	大きさと計算のしかたのちがいが、同じところ

本文は数式を読まなくても分かるように書きました。数式は脚注と付録Aにまとめてあります。同じ4つの名前で、学習後の 268 個の数（重み）の意味を読み解いた冊子『小さなAIの頭の中をのぞく』や、元の設定で 200 行を順に読む冊子『microgpt を読む』とあわせて読むと、理解が深まります。

- ・ プログラム本体：<https://gist.github.com/karpathy/8627fe009c40f57531cb18360106ce95>
- ・ 作者による解説（英語）：<https://karpathy.github.io/2026/02/12/microgpt/>

目次

1	全体像と準備	3
2	1文字を読む	9
3	外れ具合と学習	17
4	名前を作る	25
5	本物の AI と比べる	31
	おわりに	36
	付録A 数式	37
	付録B 用語のまとめ	40
	付録C 数値の求め方とプログラムの入手先	41

1 全体像と準備

1.1 大きな AI も「次を当てる」をくり返すだけ

大きな AI は、これまでの文章の続きとして、次に来る言葉を一つ予想します。「むかし むかし ある」まで読んだら次は「ところ」, 「むかし むかし ある ところ」まで読んだら次は「に」...と, 1つ当てたら付け足して, また次を当てます¹. スマートフォンの「予測変換」と同じ考え方で, ちがうのは, ずっと長い文章を読んで, ずっと上手に予想できることです.

microgpt も同じことをします. ただし材料は文章ではなく名前で, 予想する単位は1つの文字です(表1). 材料と単位が小さいだけで, やっていることは同じです. 名前なら全体を目で見て確かめられるので, しくみを調べるのにちょうどよい大きさです.

表1 大きな AI と microgpt

	大きな AI (LLM)	microgpt
学ぶ材料	大量の文章	英語の名前 (この冊子では4つ)
1つの文書	1つの記事や本	1つの名前 (emma など)
予想する単位	トークン (単語のかけら)	1つの文字
作るもの	文章の続き	新しい名前

1.2 microgpt : 約200行でできた小さな AI

microgpt は, NumPy や PyTorch のような特別なライブラリを使わず, Python に最初から入っている機能だけで書かれています. プログラムの冒頭には「このファイルがアルゴリズムのすべてで, それ以外は効率化にすぎない」という趣旨のことが書かれています. 本物の AI とのちがいは主に大きさと速さで, 計算の中身は同じだ, という主張です. 5章でこの主張を確かめます.

元の microgpt は約3万2千個の名前で学びますが, この冊子では4つの名前に減らした「小さい例」を使います(表2). プログラムは書きかえていません. 数字を4か所変えて, 学ぶ名前を4行にただけです². 小さくすると, AI の中の数をすべて紙の上で読めます.

¹大きな AI が1回に予想する単位は, 単語そのものではなく, 単語をさらに分けた「トークン」と呼ばれるかけらです. 日本語では1~2文字くらいのことが多いです.

²76行目 `n_embd 16→4`, 77行目 `block_size 16→5`, 78行目 `n_head 4→1`, 152行目 `num_steps 1000→300`. 学ぶ名前は `input.txt` というファイルに1行に1つずつ書きます. 冊子『microgpt を読む』5節の小さい例と同じ設定です.

表2 元の microgpt と、この冊子の小さい例

	元の設定	小さい例
学ぶ名前の数	32,033	4 (emma, ava, mia, anna)
文字の種類（+区切りの印）	27	7
1文字を表す数の個数	16	4
一度に読める長さ	16	5
Attention の見方の数	4	1
練習の回数	1,000	300
つまみ（重み）の数	4,192	268
計算の時間	2～3分	約2秒

1.3 プログラムは6つの部分でできている

プログラムは、上から順に次の6つの部分でできています（表3、行番号は元の microgpt.py の 200行のもの）。

表3 プログラムの6つの部分

部分	行	すること
1 データ	14-21	名前を読み込み、順番をシャッフルする
2 トークナイザ	23-27	文字を番号に変える規則を作る
3 自動微分	29-72	「どのつまみを、どちらに回せばよいか」を計算するしくみ
4 つまみとモデル	74-144	つまみ（重み）を用意し、次の文字の得点を出す本体
5 学習	146-184	外れ具合を測り、つまみを少しずつ回す
6 生成	186-200	学習したモデルで新しい名前を作る

つまみ（重み）とは AIの中には268個の数があり、これを**重み（weight）**または**パラメータ（parameter）**と呼びます。ラジオの音を合わせるつまみのように、一つ一つ回して値を変えられる数です³。最初はでたらめな値（小さな乱数）で、練習をくり返すたびに、当たりやすくなる向きへ少しずつ回します。練習が終わったときのつまみの位置が、AIが「覚えたこと」のすべてです。268個のつまみは9つの表（行列）に分けて並んでいます。表の中身と使い方は2章、つまみの回し方は3章で説明します。

練習のくり返し 練習は、部活の練習にたとえると分かりやすいでしょう（図1）。問題を解き（4モデルで次の文字を予想する）、採点し（損失で外れ具合を測る）、原因を調べ（3自動微分で、どのつまみがどれだけ悪かったかを求める）、フォームを直す（5Adamで、つまみを少しだけ回す）。1つの名前を1周し、これを300周くり返します。

³この冊子の「つまみ」は、冊子『microgptを読む』と同じたとえです。冊子『小さなAIの頭の中をのぞく』では、表に並んだ「重み」と呼んでいます。どちらも同じものです。

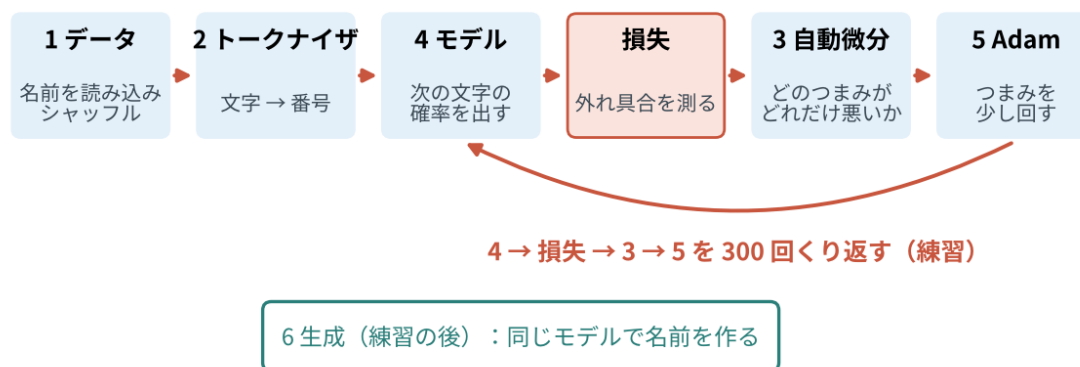


図1 プログラムの部分と、練習のくり返し。番号は表3の番号。

練習と本番 同じモデルを2つの場面で使います。練習（学習）では、正解の名前を見ながら1文字ずつ次を予想し、外れ具合に応じてつまみを回します。本番（生成）では正解はなく、始まりの印から1文字ずつ予想し、確率に従って文字を選んで付け足します。「終わり」が選ばれたら名前の完成で、つまみはもう回しません。予想の計算（4モデル）は、練習でも本番でもまったく同じです。

1.4 学習データ：4つの名前

学習には emma, ava, mia, anna の4つの名前を使い、1つの名前を1つの「文書」として扱います。この4つを選んだのには理由があります。

- ・ **文字の種類が少ない**：a, e, i, m, n, v の6種類だけなので、AI 中の表が小さく、全部を読めます。
- ・ **「正しく迷う」場面がある**：始まりの次（a か e か m）と、a の次（v か n）は、データ自体が答えを一つに決めていません。
- ・ **同じ役目の文字がある**：mia の i と ava の v は、どちらも「次は必ず a」です。AI がこれに気づくかどうかを確かめられます。
- ・ **どれも4文字以内**：一度に読める長さは5なので、最後の「終わり」まで練習できます。

ばらばらの名前（たとえば sophia, mary, celine, elena）にすると、始まりの文字以外は次が一つに決まってしまう、丸暗記しか見えなくなります。文字を共有する名前を選んだのはそのためです。

シャッフル プログラムは、読み込んだ名前の順番をまずランダムに並べかえ（シャッフル）、その順に1つずつ練習に使います。乱数の種 42 では、順番は mia, ava, anna, emma になります。最後まで来たら先頭に戻るの、300回で、どの名前もちょうど75回ずつ練習します⁴。

⁴コンピュータの乱数は「乱数の種」と呼ばれる出発点の数から計算で作られ、同じ種からは毎回同じ数の列が出ます。microgpt は種を 42 に決めている（12行目）ので、誰が動かしても1回目は mia です。種を変えたと、シャッフルの順、つまみの最初の値、名前を作るときにくじ引きがすべて変わります。

1.5 文字の種類を集めて、番号をふる

AI は文字をそのまま扱えないので、文字に番号をつけます。4つの名前に出てくる文字を集めて、重なりを除き、アルファベット順に並べ、0 から番号をふります。最後に、区切りの印 BOS（ビーオーエス, beginning of sequence）に次の番号 6 をふります（表4）⁵。

表4 文字と番号

文字	a	e	i	m	n	v	BOS
番号	0	1	2	3	4	5	6

学ばせる名前を変えれば、文字の種類も番号も変わります。元の約3万個の名前なら a~z の 26種類+BOS の27種類です。名札にない文字（b や z など）は、この AI にとって存在しません。

BOS：始まりと終わりを示す1つの印 名前を練習するとき、前と後ろに BOS を付けます。前の BOS は「ここから名前が始まる」という合図で、AI は BOS を読んで名前の最初の文字を当てます。後ろの BOS は「ここで名前が終わる」という答えで、AI が BOS を予想したら名前の完成です。名簿の名前と名前の間にある区切り線のように、1本の線が上の名前の終わりとの下の名前の始まりを同時に示しています。

4つの名前を番号の列にする こうして、4つの名前は次の番号の列になります⁶。ここから先の計算は、すべてこの番号の列で行います。

emma	→	[6, 1, 3, 3, 0, 6]	ava	→	[6, 0, 5, 0, 6]
mia	→	[6, 3, 2, 0, 6]	anna	→	[6, 0, 4, 4, 0, 6]

AI が知っているのは番号だけ：AI は a の形も音も知りません。すべての文字の呼び名を一度に入れかえても（a を「赤」、e を「青」...）、学ぶ中身はまったく同じです。サッカーの記録に背番号しか書かれていなくても、「7番から10番にパスが出やすい」ことは記録から学べるのと同じです。文字を単位にするのは microgpt の選び方で、大きな AI は、よく出る文字の組をまとめた「トークン」を単位にしています。

1.6 練習問題を作る

1つの名前からは、「ここまで読んだら、次は何？」という問題がいくつもできます。emma からは表5の5問です。読んだところは、いつも名前の頭からです。AI は前を見ることはできませんが、先（答え）を見ることはできません。最後の BOS は答えになるだけで、それを読んだで次を当てる問題はありません⁷。

⁵プログラムでは `uchars = sorted(set(''.join(docs)))`（24行目）で文字を集めて並べ、`BOS = len(uchars)`（25行目）で BOS の番号を決めます。この「文字と番号を行き来する規則」をトークナイザと呼びます。

⁶プログラムでは `tokens = [BOS] + [uchars.index(ch) for ch in doc] + [BOS]`（157行目）です。`uchars.index(ch)` は「文字 ch が何番目か」を調べる命令です。

⁷1回の練習では、1つの名前の問題を全部（emma なら5問）解いて、その平均の外れ具合でつまみを回します。

表5 emma から作る練習問題

問題	読んだところ	次の文字（正解）
1	始まり（BOS）	e
2	e	m
3	em	m
4	emm	a
5	emma	終わり（BOS）

4文字の名前は5問、3文字の名前は4問なので、4つの名前で $5 + 4 + 4 + 5 = 18$ 問です。このうち「始まり」は4つの名前すべてに、「a」は ava と anna の両方に出てくるので、同じ読んだところをまとめると **14の場面**になります。

名前の木 4つの名前を1本の木にまとめると、図2ようになります。左の BOS から右へたどると、4つの名前のどれかになります。枝が分かれる場所は2か所だけです。始まりの次と、a の次です。ほかの12の場面では、次の文字が一つに決まります。

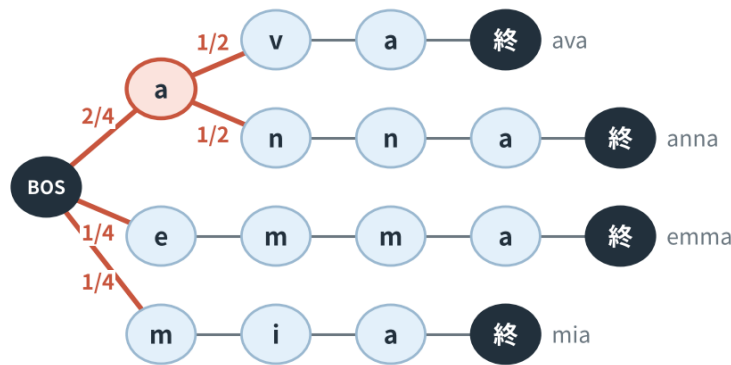


図2 4つの名前の木。「終」は終わりの BOS. 赤い数字は、データの中でその枝に進む割合。

1.7 正しく迷う：答えが一つに決まらない場面

始まりの次は、4つの名前のうち a が2つ（ava, anna）, e と m が1つずつです。a の次は v（ava）と n（anna）が1つずつです（図3）。こうした場面では、どれか一つに決めるより、「a が 50%, e が 25%, m が 25%」とデータの割合どおりに答えるのが最もよい答えです⁸。くじ引きの箱に、a を2本、e と m を1本ずつ入れておくようなものです。

⁸ 「最もよい」とは、3章で説明する罰点（損失）の平均がいちばん小さくなる、という意味です（付録A.1）。

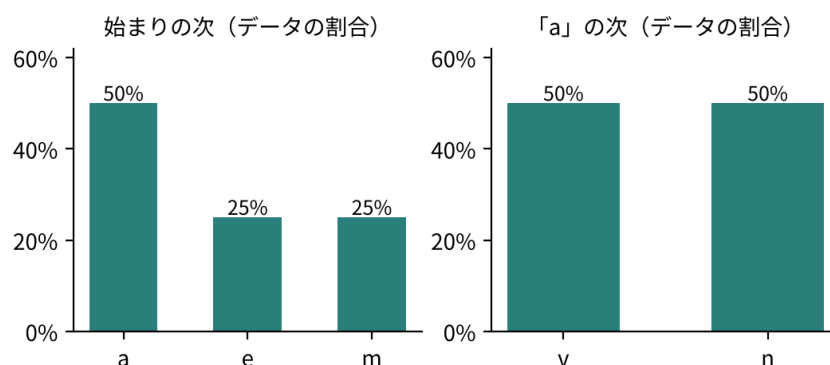


図3 答えが分かれる2つの場面（データの割合）。

どんなに練習しても、迷いは0にならない：18問のうち12問は答えが一つに決まるので、迷いは始まりとaの次の6問だけから来ます。「平均して何択で迷っているか」で表すと、練習の前は7択（7種類から当てずっぽう）、データから決まる限界は1.36択です^a。AIがどんなに賢くなっても、1.36択より小さくはなりません。300回の練習の後は1.41択で、この限界にかなり近づきます（3章）。

^a正確には、正解の文字につけた確率の「かけ算の平均」（幾何平均）の逆数です。限界の1.36は罰点の平均0.308から求めました（付録A.1）。

1.8 人間が前もって決める設定

料理の「何人分を作るか」「何分煮るか」のように、作り始める前に決める数があります（表6）。つまみ（重み）とちがい、練習しても変わりません。専門用語ではハイパーパラメータ（hyperparameter）と呼びます。一度に読める長さが5なので、5文字以上の名前は後ろが切られ、最後の「終わり」まで練習できません。4つの名前がどれも4文字以内なのはそのためです。

表6 人間が前もって決める設定

設定	プログラムの名前	小さい例	元	意味
1文字を表す数の個数	n_embd	4	16	2章の「黒板」の幅
一度に読める長さ	block_size	5	16	始まり+4文字まで
Attentionの見方の数	n_head	1	4	2章で説明
練習の回数	num_steps	300	1000	各名前75回
つまみを回す大きさ	learning_rate	0.01	0.01	3章で説明
乱数の種	random.seed	42	42	同じ種なら同じ結果

1章のまとめ：AIは「次を当てる」をくり返しています。プログラムは6つの部分からなり、練習（予想→採点→原因を調べる→つまみを回す）を300回くり返したあと、本番で名前を作ります。名前は番号の列になり、18問の練習問題になります。答えが分かれるのは始まりの次とaの次だけで、そこでは正しく迷うのが最善です。

2 1文字を読む

この章では、AI が1文字を読んで次の文字の確率を出すまでの計算を、実際の数でたどります。使うのは、300回の練習を終えたつまみ（学習後の重み）です。予想がよく当たるので、計算の意味が見えやすいからです。つまみをどう回したか（学習）は3章で見ます。

2.1 1回の予想の流れ：5つの段階

1文字を読むたびに、AI は図4の5つの段階で計算します⁹。この5段階を、5つの位置（BOS, e, m, m, a）それぞれで行います。どの位置でも同じつまみを使います。

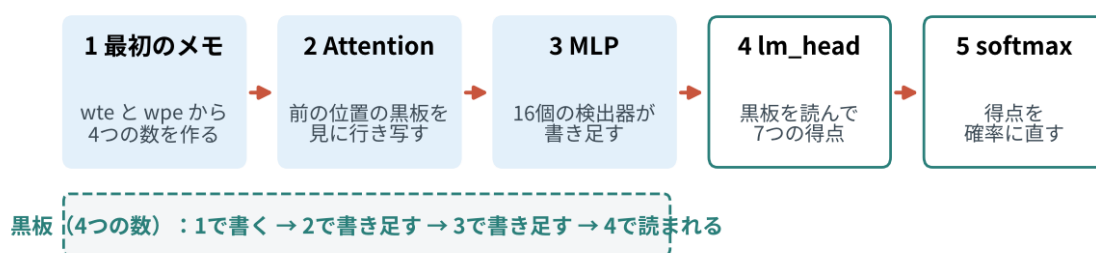


図4 1回の予想の5つの段階.

2.2 AI 中の黒板

文字の位置ごとに、4つの数を書ける黒板が1枚ずつあります。emma なら5枚です（図5）。まず最初のメモを書き、Attention と MLP が結果を足し算で書き足します。前に書いたものは消しません。最後に lm_head が黒板を読んで、次の文字の得点にします¹⁰。

BOS	e	m	m	a
p0	p1	p2	p3	p4
-0.15	+0.11	-0.15	-1.42	-0.30
+1.89	+0.13	-0.10	+0.31	-1.15
+0.53	-1.47	-1.42	-0.07	+1.46
+0.38	+1.34	+1.39	-1.38	-0.68

図5 5枚の黒板に書かれた最初のメモ（学習後）。赤は正、青は負の数.

黒板と、つまみの表はちがうもの 黒板は1回の予想の中で書き足され、次の名前を読むときは新しい黒板で始めます。練習の後に残りません。一方、9つの表（つまみ）は練習1回ごとに回され、練習の後に268個の重みとして残ります。黒板は1問ごとの計算用紙、表は練習のたびに少しずつ書き直していく自分のノートです。この章で追いかけるのは計算用紙（黒板）の方で、ノート（表）は学習後のものに固定しておきます。

⁹プログラムでは、1～4が `gpt()` という関数（108–144行目）、5が `softmax()`（97–101行目）です。

¹⁰専門用語では、この黒板を「残差の流れ」（**residual stream**）と呼びます。書き足しは $x \leftarrow x +$ （計算の結果）という足し算で、プログラムでは `x = [a + b for a, b in zip(x, x_residual)]`（134行目、141行目）です。

追いかける位置：emmaの2つ目のm emmaの「emm」まで読んだ位置(p3)を追いかけます。同じmを読んでいるのに、1つ目のm(p2, 読んだところem)では次がm, 2つ目のm(p3)では次がaです。文字だけでは決まらない問題で、「何文字目か」と「前に何があったか」を使わないと解けません。AIはこの2つを見分けて、p3でaに98.8%をつけました。どの段階がそれを使うのかに注目してください。

2.3 段階1：最初のメモ

wte：文字ごとに4つの数が並んだ辞書 wteは、7種類の文字それぞれに4つの数を割り当てた表です(図6左)。辞書を引くように、今読む文字の行を取り出します。mを読むなら、mの行(-0.06, +0.03, -0.12, +0.00)です。emmaの2つのmは、どちらも同じmの行を使います。wteだけでは、2つのmは区別できません。4つの数の値は練習で決まったもので、1つずつの数には決まった意味がありません¹¹。

wte (7行×4列)					wpe (5行×4列)				
	d0	d1	d2	d3		d0	d1	d2	d3
a	+0.12	-0.21	+0.19	+0.18	p0	+0.05	+0.21	+0.14	+0.08
e	-0.17	+0.07	-0.08	+0.28	p1	+0.18	-0.06	-0.06	-0.14
i	-0.13	+0.05	+0.05	-0.23	p2	+0.05	-0.04	-0.04	+0.15
m	-0.06	+0.03	-0.12	+0.00	p3	-0.22	+0.03	+0.10	-0.28
n	-0.02	+0.01	-0.01	-0.01	p4	-0.18	-0.01	+0.09	-0.31
v	-0.13	+0.06	+0.05	-0.22					
BOS	-0.09	+0.23	-0.02	+0.01					

図6 学習後のwte(左)とwpe(右)。枠は、p3で使うmの行とp3の行(灰色はp2の行)。

wpe：位置ごとに4つの数が並んだ表 wpeは「何文字目か」に応じた4つの数の表です(図6右)。p0は始まり(BOS)の位置、p1は1文字目、..., p4は4文字目です。1つ目のmはp2の行、2つ目のmはp3の行を使います。ここで初めて2つのmに差がつきます。リレーの走者のようなものです。同じ選手でも、第2走者か第3走者かで仕事がちがいます。位置の表があるのは、Attentionには順番を見分ける力がないからです¹²。

文字の数と位置の数を足す 2つの表の同じ欄(d0 どうし、d1 どうし...)を足します(図7)。p2の合計(-0.02, -0.01, -0.15, +0.15)と、p3の合計(-0.28, +0.06, -0.01, -0.28)はまったくちがう数になりました。文字の情報(m)と位置の情報(3文字目)が、1組の4つの数に混ざって入ります。

¹¹d0~d3は4つの数の欄(4次元の空間の4本の軸)です。軸の向きを回してもAIの働きは変わらないので、「d2が負だから...」とは読めません。くわしくは冊子『小さなAIの頭の中をのぞく』3~4節。wteはword token embedding(トークンの埋め込み)の略です。

¹²Attentionは前の位置をまとめて「袋に入れて」見渡す計算なので、並び順の情報を自分では持てません。wpeはword position embedding(位置の埋め込み)の略です。

	wte の m の行					wpe の位置の行					合計			
p2 の m	-0.06	+0.03	-0.12	+0.00	+	+0.05	-0.04	-0.04	+0.15	=	-0.02	-0.01	-0.15	+0.15
p3 の m	-0.06	+0.03	-0.12	+0.00	+	-0.22	+0.03	+0.10	-0.28	=	-0.28	+0.06	-0.01	-0.28

図7 wte の行と wpe の行を足す (p2 と p3 の m) .

大きさをそろえる (rmsnorm) 足した4つの数は、最後に大きさをそろえてから黒板に書きます. p3 の場合, (1) 4つの数をそれぞれ2乗して平均すると $\{(-0.28)^2 + 0.06^2 + (-0.01)^2 + (-0.28)^2\} \div 4 = 0.0402$, (2) その平方根は 0.200, (3) 4つの数をそれぞれ 0.200 で割ると $(-1.42, +0.31, -0.07, -1.38)$ になります. これが p3 の最初のメモです.

割ったあとの4つの数は、2乗の平均がいつも 1 になります. 数の並びの向きは変わらず、大きさだけがそろいます¹³. こうしておく、このあと黒板を読む係が、数の大きさにまどわされず、並び方 (向き) だけで判断できます. 録音の音量の自動調整のように、大声でも小声でも同じ音量にそろえ、話の中身を聞き取りやすくするのです.

最初のメモを比べる 同じ m でも、p2 と p3 の最初のメモは向きがまったくちがいます (向きのそろい方 cos は -0.41)¹⁴. 反対に、ちがう文字の e (p1) と m (p2) は、ほとんど同じ向きのメモになりました (cos は 0.98). どちらも「次は m」だからです. メモの近さは、文字の形ではなく「次に何が来るか」という役目の近さを表すように、練習で作られています.

2.4 段階2: Attention

図書館で調べものをするしくみ Attention は、前の位置の黒板を見に行き、必要なものを写してくるしくみです. 図書館にたとえると、各位置の黒板は本で、それぞれ「見出し」 (key, k) と「中身」 (value, v) を持っています. 今の位置は「何を探しているか」のメモ (query, 問い合わせ, q) を作り、次の3つを行います.

1. 問い合わせ q と、各位置の見出し k を比べる. 似ている見出しほど高い得点にする.
2. 得点を割合に直す (合計 100%). どの本をどれだけ読むかの配分を決める.
3. 割合に応じて中身 v を混ぜて写し、自分の黒板に書き足す.

見に行けるのは、自分と自分より前の位置だけです. 先 (答え) の文字はまだ見えません. p3 なら BOS, e, m, m の4つの黒板を見られます.

q, k, v を作る 黒板の4つの数に、3つの表 wq, wk, wv をそれぞれ掛けて、q, k, v を作ります¹⁵. p3 の問い合わせは $q = (+1.33, +0.38, -1.19, +1.44)$ です. k と v は、それぞれの位

¹³rmsnorm は root mean square normalization (2乗平均の平方根でそろえる) の略です (103-106行目). 矢印の長さで言えば、4つの数ならいつも長さ 2 ($= \sqrt{4}$) になります. 0 で割らないように、実際は平均に 0.00001 を足してから平方根をとります (付録A.2).

¹⁴cos (コサイン) は、2つの数の並びを矢印とみたときの向きのそろい方です. 同じ向きなら +1, 直角なら 0, 反対向きなら -1 です (付録A.2).

¹⁵「表を掛ける」は、表の各行と黒板の4つの数を成分ごとに掛けて足す計算です (行列とベクトルの積). 例: q の1つ目 = wq の1行目と黒板の積 = $(-0.41) \times (-1.42) + (+0.64) \times (+0.31) + (+0.12) \times (-0.07) + (-0.40) \times$

置を読んだときに作って、あとの位置のためにとっておいたものです¹⁶。

問い合わせと見出しの似かたを測る q と各位置の k を、成分ごとに掛けて足し（積の和）、2 で割ったものが得点です（表7）¹⁷。BOS の見出しとの積の和は $(+1.33)(+1.27) + (+0.38)(+0.09) + (-1.19)(-1.19) + (+1.44)(+1.56) = +5.38$ です。4つの数の符号がそろっているほど、積の和は大きくなります。 q と BOS の見出しは、4つとも同じ符号でした。得点がいちばん高いのは BOS（始まり）で、 p_3 の m は名前の始まりの黒板を見に行こうとしています。

表7 p_3 の問い合わせと各位置の見出し

見に行く位置	見出し k	積の和	$\div 2 =$ 得点	e の得点乗	割合
BOS (p_0)	$(+1.27, +0.09, -1.19, +1.56)$	$+5.38$	$+2.69$	14.76	76.2%
e (p_1)	$(-0.09, +1.17, +0.11, -0.29)$	-0.22	-0.11	0.90	4.6%
m (p_2)	$(-0.03, +1.26, +0.06, -0.26)$	$+0.00$	$+0.00$	1.00	5.2%
m (p_3 , 自分)	$(+0.56, -0.16, -0.55, +0.45)$	$+1.99$	$+1.00$	2.71	14.0%
合計				19.36	100%

得点を割合に直す (softmax) e (約 2.718) の得点乗を求めて、合計で割ります（表7の右2列）。得点が1 高いと、約 2.7 倍多く見ます。 p_3 の m は、注目の 76% を名前の始まり（BOS）に向けました（図8左）。持ち時間の配分のように、得点の高い本ほど長く読みますが、ほかの本もほんの少しは読みます¹⁸。

中身を混ぜて写し、黒板に書き足す 割合に応じて、4つの位置の中身 v を混ぜます。 $76.2\% \times v(\text{BOS}) + 4.6\% \times v(e) + 5.2\% \times v(m) + 14.0\% \times v(m) = (-1.11, +0.76, -0.18, -1.18)$ です。それを w_0 の表で黒板の形に直して¹⁹ $(-0.97, +2.22, -0.58, -0.13)$ とし、最初のメモに足します。黒板は $(-2.38, +2.53, -0.65, -1.51)$ になりました。写してきたものの大部分は、名前の始まりの黒板の中身です。「始まりから3文字目の m 」という情報が、 p_3 の黒板に書き加えられ、このあとの MLP がこれを手がかりにできます。

$(-1.38) = +1.33$ 。4行の表なので4つの数ができます。

¹⁶121–122行目。前の位置の k と v をとっておくしくみは、大きな AI で KV キャッシュと呼ばれるものと同じです（4章）。

¹⁷2 で割るのは、数の個数（4）の平方根です。数の個数が多いと積の和が大きくなりすぎるのを防ぎます（付録 A.3）。

¹⁸softmax は、得点の差を「何倍か」に直して、合計が1 になるように配る計算です。最後の段階5でも同じ計算を使います（付録A.3）。

¹⁹ w_0 の掛け算は、混ぜた中身を黒板の4つの欄に合わせて書き直す計算です（133行目）。 w_q, w_k, w_v, w_0 の4つの表を合わせて Attention の部品です（各 $4 \times 4 = 16$ 個、計64個のつまみ）。

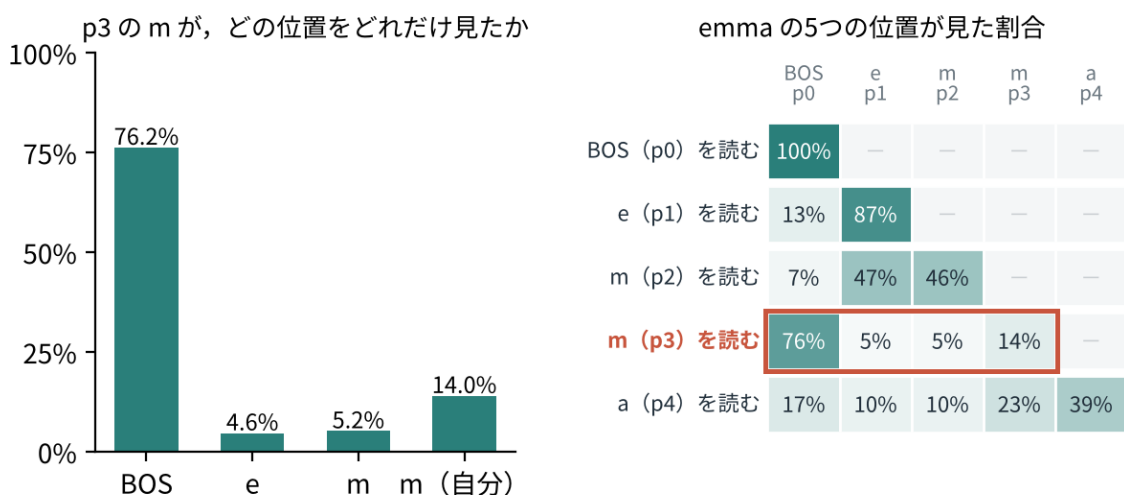


図8 Attention の割合. 左: p3 の m が見た割合. 右: emma の5つの位置が見た割合 (各行の合計100%. 「—」はまだ読んでいない先の位置).

emma の5つの位置は、どこを見たか 図8右を見ると、右上の「—」は、まだ読んでいない先の位置で、見ることはできません. 始まり (p0) は自分しか見られないので、いつも100%です. そのため始まりの予想は、どの名前でもまったく同じ計算になります. p3 の m だけが、始まり (BOS) を強く見えています. 4つの名前すべてで、次が a になる場面 (mi, av, ann, emm) では、BOS に76~84% 注目していました.

2.5 段階3: MLP

「型紙」と照らし合わせる検出器の集まり MLP (multi-layer perceptron, 多層パーセプトロン) は、黒板の4つの数を一度16個に広げてから、また4つに戻す部分です²⁰. fc1 の16本の行は、それぞれ1枚の「型紙」です. 黒板の並び方が型紙に似ていれば反応 (点灯) し、似ていなければ黙ります. fc2 は、点灯した検出器ごとに「これを黒板に足せ」という内容を持っています. MLP は自分の黒板だけを見ます. ほかの位置の情報は、すでに Attention が写ってきています. 波形を型と照らし合わせ、基準を超えたら知らせる地震計のトリガーと同じ形です.

16個の検出器の反応 型紙と黒板の積の和 (似かた) を16個求めます (図9). 似かたが負のもの (h12, h14) は0にして黙らせます. この「負なら0」を **ReLU** と呼びます²¹. p3 では14個が点灯しました. 次が a になる場面 (mi, av, ann, emm) では、14~15個が一斉に点灯します. 「もうすぐ a」のときの共通の反応です.

²⁰137-141行目. 読む前に、ここでも大きさをそろえます (rmsnorm).

²¹ReLU は **rectified linear unit** の略です. 神経細胞の「刺激の合計がしきい値を超えると発火する」しくみをまねたもので、ここではしきい値が0です (付録A.4).

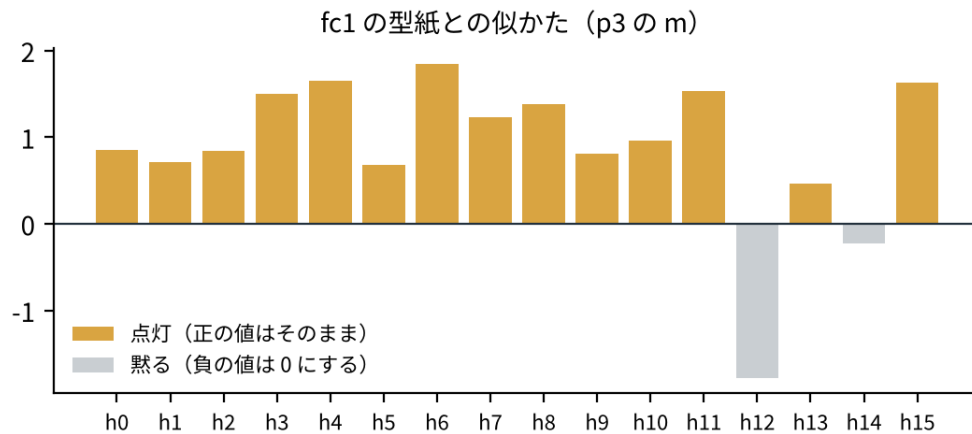


図9 p3 の m での、16個の型紙との似かた。負の値（灰色）は0にされる。

点灯した検出器が、黒板に書き足す 点灯の強さに応じて fc2 の内容を足し合わせると、 $(-4.74, +7.97, +0.76, -2.59)$ になります。これを黒板に足すと、最後の黒板は $(-7.12, +10.50, +0.11, -4.10)$ です。黒板に書いたものの大きさ（矢印の長さ）を比べると、最初のメモが2.00、Attentionの結果が2.49、MLPの結果が9.66でした²²。いちばん大きく書き足したのはMLPです。つまみの約半分（128個）がMLPにあり、「こういうときは次にこれ」という規則の多くをここで持っています。

2.6 段階4・5：得点と確率

lm_head：最後の黒板を読んで得点を出す lm_headの各行は、その文字の「読み取り用の型紙」です。最後の黒板と行の積の和が、その文字の得点になります²³。たとえばaの得点は

$$(-0.38) \times (-7.12) + (+0.34) \times (+10.50) + (+0.05) \times (+0.11) + (-0.35) \times (-4.10) = +7.70$$

です²⁴。7つの文字の得点は、a +7.70, e +3.13, i -2.39, m +0.80, n -9.62, v -6.36, 終わり +0.89で、aがとびぬけて高くなりました（図10左）。

得点を確率に直す (softmax) Attentionのときと同じ softmax で、得点を確率に直します。aとeの得点の差は4.57で、eの4.57乗は約96なので、aはeの約96倍ありそう、という答えになります。正解のaに98.8%をつけました（図10右）。

²²矢印の長さは、4つの数の2乗の和の平方根です。

²³lm_head は language model head（言語モデルの頭部、最後の出力の部分）の略です（143行目）。得点は専門用語でロジット（logit）と呼びます。

²⁴表の数を小数2桁に丸めて計算すると、最後の桁が少しずれることがあります。

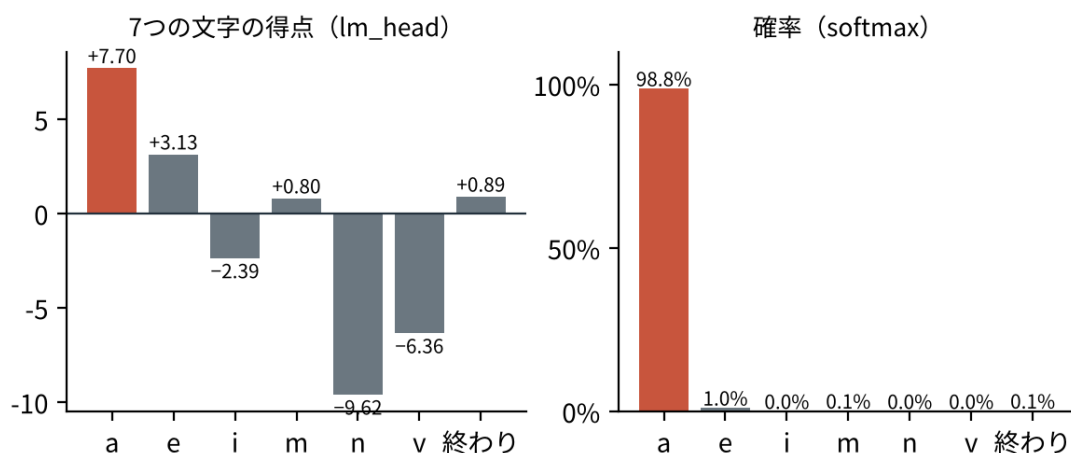


図 10 p3 (emm の次) の7つの得点 (左) と確率 (右) .

得点は3つの書き込みの足し算に分けられる 黒板への書き込みはいつも足し算で, lm_head も掛けて足すだけの計算なので, 得点もぴったり3つに分かれます (図11) ²⁵.

a の得点 +7.70 = 最初のメモ +1.12 + Attention +1.13 + MLP +5.44

p3 では, 最初のメモと Attention が a を少し押し上げ, MLP が大きく押し上げました. n と v は MLP が大きく押し下げています.

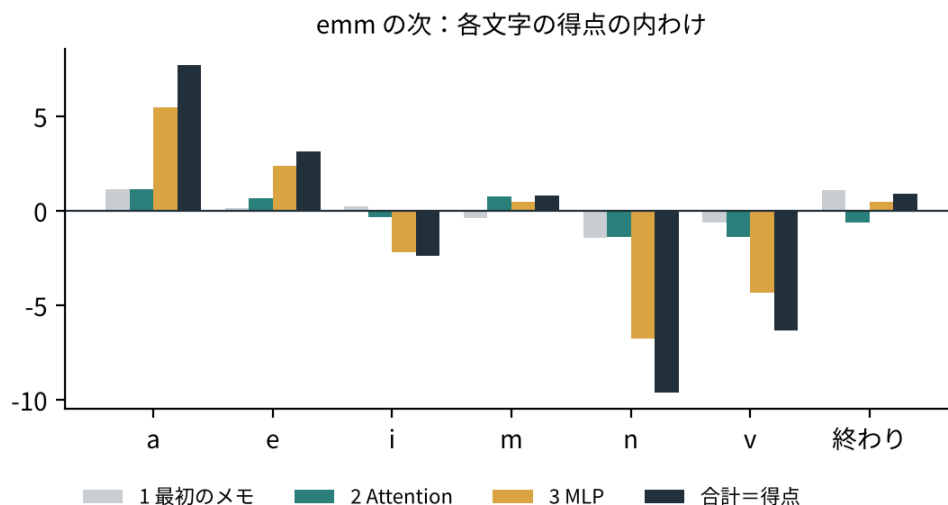


図 11 emm の次の, 各文字の得点の内わけ.

名前の終わりでは, MLP が「終わり」を押し上げる emma を読み終えた p4 では, 「終わり」の得点は +12.26 = 最初のメモ +1.57 + Attention +0.54 + MLP +10.15 で, 確率は 99.6% でした. 点灯した検出器は9個で, p3 とは組み合わせがちがいます. 「a を読み終えた」場面に反応する組が点灯しています. ここでも決め手は MLP です.

²⁵lm_head の前には大きさをそろえる計算 (rmsnorm) がないので, 分けた3つの和は得点と小数第14位まで一致します (付録A.5) .

2.7 emma の5問と、14の場面

emma の5つの問題の答えは表8のとおりです。p1～p4 は、正解に 94～99.6% をつけました。始まり (p0) では e は 19.3% しかありませんが、これは失敗ではありません。データでは始まりの次は a が 50%, e と m が 25% ずつで、AI もほぼその割合で迷っています (1章「正しく迷う」)。

表8 emma の5つの問題の答え

位置	読んだところ	正解	正解の確率	AI の予想 (上位3つ)
p0	始まり	e	19.3%	a 50.9%, m 29.7%, e 19.3%
p1	e	m	94.0%	m 94.0%, e 3.9%, a 1.3%
p2	em	m	94.4%	m 94.4%, e 3.7%, a 1.2%
p3	emm	a	98.8%	a 98.8%, e 1.0%, 終わり 0.1%
p4	emma	終わり	99.6%	終わり 99.6%, a 0.4%

4つの名前の14の場面すべてで見ると (表9), 答えが一つに決まる12の場面では 80～99.8%, 迷う2つの場面 (始まり, a) ではデータの割合に近い確率をつけています。いちばん低いのは m の次の i (80.2%) でした。

表9 14の場面での、正解の確率 (学習後)

読んだところ	データ	AI
始まり	a 50%, e 25%, m 25%	a 50.9%, e 19.3%, m 29.7%
m	i 100%	i 80.2%
mi	a 100%	a 99.2%
mia	終わり 100%	終わり 99.7%
a	v 50%, n 50%	v 49.9%, n 45.6%
av	a 100%	a 99.2%
ava	終わり 100%	終わり 99.8%
an	n 100%	n 84.4%
ann	a 100%	a 98.8%
anna	終わり 100%	終わり 99.8%
e	m 100%	m 94.0%
em	m 100%	m 94.4%
emm	a 100%	a 98.8%
emma	終わり 100%	終わり 99.6%

2章のまとめ: 1文字を読むたびに、最初のメモ (wte+wpe, 大きさをそろえる) → Attention → MLP → lm_head の得点 → softmax の確率, の5つの段階で計算します。同じ m を見分けたのは、位置の表 (wpe) と Attention です。wpe で「何文字目か」が入り、Attention が「始まりから何文字目か」を写してきました。得点は3つの書き込みの足し算で、いちばん大きく押し上げたのは MLP でした。

3 外れ具合と学習

2章では学習後のつまみを使いました。この章では、つまみを学習前のでたらめな値に戻し、そこから300回の練習で2章のつまみにたどり着くまでを見ます。練習1回の流れは、予想する（2章の計算）→採点する（損失）→原因を調べる（勾配）→つまみを回す（Adam）でした。1回の練習では名前を1つだけ使い、mia, ava, anna, emma, mia... と4つの名前を順にくり返します²⁶。

3.1 学習前の AI

268個のつまみは、最初はすべて小さな乱数です。だいたい $-0.16 \sim +0.16$ の範囲に散らばっています²⁷。どの文字の行も、どの位置の行も、まだ何の役目も持っていません。入部したての部員のように、やる気はあっても、まだ何も知らない状態です。

つまみがでたらめなので、AI はどの問題でも7文字に 12~17% くらいずつ確率を配ります（図12）。7文字に同じ確率をつけると、どれも $1/7 \approx 14.3\%$ です。これは当てずっぽうと同じで、ここから練習が始まります。

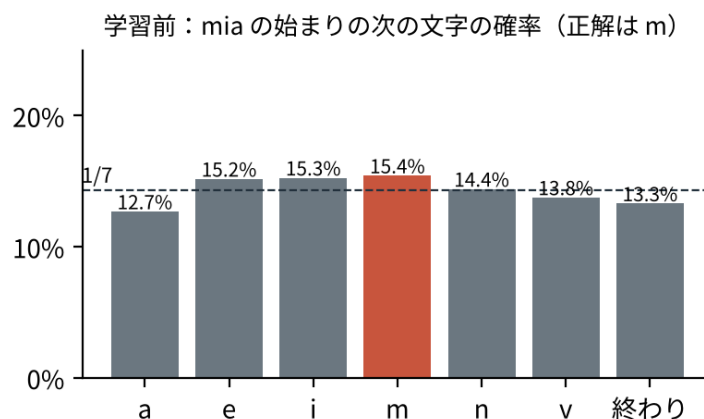


図 12 学習前のつまみで、mia の始まりの次の文字の確率。点線は $1/7$ 。

3.2 外れ具合を測る（損失）

AI は次の文字を一つに決めず、7つの文字それぞれに確率を割りふります。そこで、正解の文字につけた確率から罰点を決めます。これを**損失**（loss）と呼びます。罰点は $-\ln$ （正解の確率）で計算します（図13）²⁸。確率 1（100%）なら罰点 0（完全に的中）、0.5 なら 0.69（二択で迷った）、 $1/7$ なら 1.95（7つから当てずっぽう）、0.01 なら 4.61（「まずない」と言って外した）です。

²⁶プログラムでは、名前の取り出しが 156–157 行目、予想と損失が 163–169 行目、勾配が 172 行目（`loss.backward()`）、つまみの更新が 174–182 行目です。

²⁷平均 0、標準偏差 0.08 の正規分布（ガウス分布）から1つずつ選んでいます（80行目の `random.gauss(0, std)`）。ほとんどの値は標準偏差の2倍（ ± 0.16 ）の中に入ります。

²⁸ $\ln$ は自然対数で、「e の何乗でその数になるか」を表します。確率が 1 に近いほど 0 に近く、0 に近いほど限りなく大きくなります。

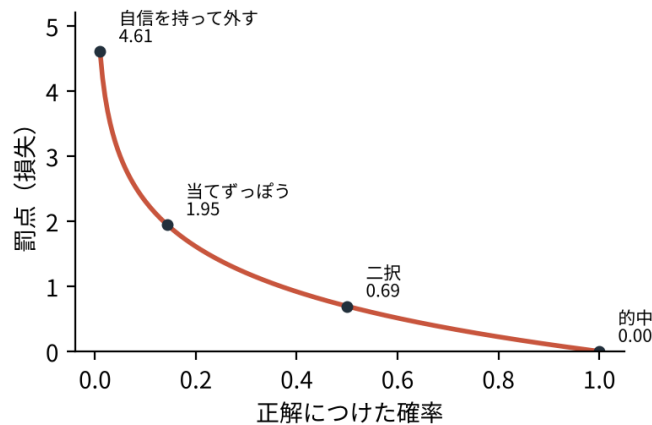


図 13 正解につけた確率と罰点.

天気予報の採点と同じ：「降水確率 70%」と言って雨なら，罰点は小さい．「1%」と言い切って雨なら，大きな減点です．自信を持って外すほど，罰点は急に大きくなります．「正解を知ったときの驚きの大きさ」とも言えます．

1回目の練習 (mia) の罰点 mia には4つの問題があります (表10)．それぞれの罰点を求めて平均したものが，1回目の練習の損失です．平均 1.953 は，当てずっぽうの罰点 ($\ln 7 \approx 1.946$) とほぼ同じです．学習の目標は，この罰点の平均を下げることです²⁹．

表 10 1回目の練習 (mia) の罰点

問題	読んだところ	正解	正解の確率	罰点	1回目後の確率
1	始まり	m	15.4%	1.869	16.5%
2	m	i	11.6%	2.151	12.3%
3	mi	a	17.1%	1.764	18.4%
4	mia	終わり	13.1%	2.029	14.0%
平均				1.953	(罰点 1.889)

なぜ「正答率」ではなく罰点なのか 正答率 (いちばん確率の高い文字が正解だった割合) は，確率を 0.15 から 0.16 に少し上げても，一番が入れかわらなければ変わりません．正答率は階段のようにとびとびにしか動かないので，「つまみを少し回したら，どれだけよくなったか」が測れません．罰点は，正解の確率がほんの少し上がるだけでも，ほんの少し下がります．なめらかに変わるので，「罰点が下がる向き」が計算できます．階段では，どちらに一步出れば下がるのか分かりませんが，なめらかな坂なら，足もとの傾きで下りる向きが分かります．

3.3 どのつまみを，どちらに回すか (勾配)

霧の山の中で谷へ下りたいとき，遠くは見えなくても，足もとの傾きは分かります．傾きの下る向きに一步步進めば，谷に近づけます (図14)．268個のつまみは，268本の方

²⁹このような損失を，専門用語では交差エントロピー損失 (cross-entropy loss) と呼びます (付録A.6)．

です。それぞれのつまみについて、「少し増やすと罰点がどれだけ増えるか」の割合を**勾配** (gradient) と呼びます。勾配が正のつまみは減らし、負のつまみは増やせば、罰点が下がります。勾配は「その場所での傾き」なので、一步進むと傾きも変わります。だから少し進んでは測り直します。

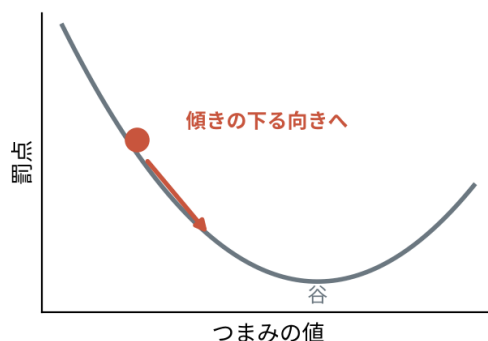


図 14 勾配は足もとの坂の傾き。

勾配を実際に確かめる `lm_head` の `i` の行の3番目のつまみ (学習前の値 0.1090) を、0.001 だけ増やしてみます。mia の罰点の平均は 1.953475 から 1.953936 に、0.000461 増えました。罰点の変化 ÷ つまみの変化 = $0.000461 \div 0.001 \approx 0.461$ が、このつまみの勾配です。プログラムが自動で計算した勾配も 0.461 で、一致します。勾配が正なので、このつまみは減らせば罰点が下がります。

ただし、268個を1つずつ回して確かめていたら、予想の計算が268回余分に必要です。大きな AI には何千億個もつまみがあります。そこで、計算のつながりを逆にたどって、全部の勾配を一度に求めます。

3.4 勾配を一度に求める (自動微分)

小さな例: $f = a \times b + a$ $a = 2$, $b = 3$ として、 $c = a \times b = 6$, $f = c + a = 8$ を計算します (図15左)。プログラムは計算をするたびに、「どの数とどの数から、どんな計算で作ったか」を記録しておきます。次に、 f から出発して記録を逆にたどります (図15右)。足し算はそのまま伝え、掛け算は「相手の値」倍して伝えます。 a には2つの道から 1 と 3 が届くので、合わせて 4 です。 b は $1 \times a = 2$ です。

確かめると、 a を 0.01 増やすと f は 8.04 になり (0.01 の4倍)、 b を 0.01 増やすと 8.02 になります (0.01 の2倍)。逆にたどって求めた 4 と 2 は、「その数を少し増やすと、 f がその何倍増えるか」を表しています。microgpt では f が罰点で、 a や b が268個のつまみにあたります。

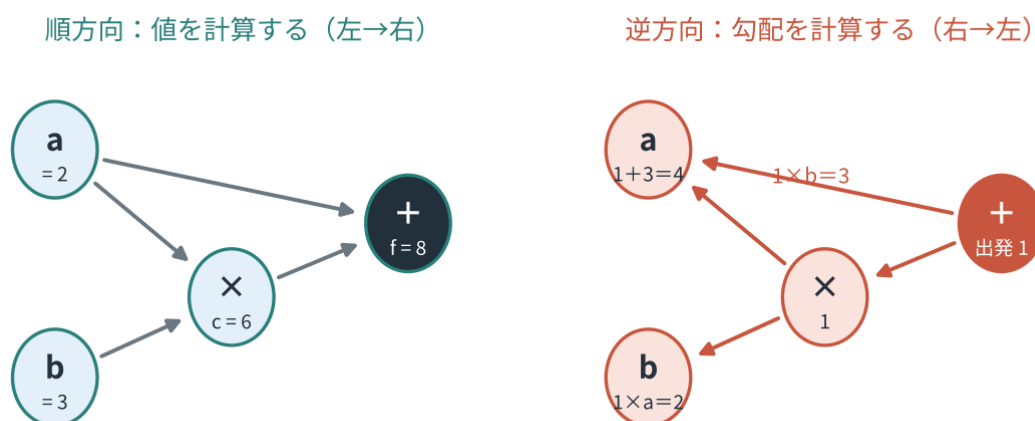


図 15 $f = a \times b + a$ の計算の記録. 左：値を計算する. 右：逆にたどって勾配を計算する.

連鎖律：歯車の回転比を掛けていく つまみを回すと黒板の数が変わり、黒板が変わると得点が変わり、得点が変わると確率が変わり、確率が変わると罰点が変わります. 影響は歯車のように順に伝わって罰点に届きます. つまみを1回すと罰点が何回あるかは、となりどうしの歯車の回転比をすべて掛け合わせれば分かります. これを微分の**連鎖律**（chain rule）と呼びます. 逆方向の計算は、罰点の側から出発して、回転比を掛けながら1段ずつさかのぼります. 途中の結果を使い回せるので、268個の勾配が1回のさかのぼりでそろいます³⁰.

得点の勾配は「確率－正解」 さかのぼりの最初の一步は、とても簡単な形になります. 各文字の得点の勾配は、その文字の確率－（正解なら 1, ちがえば 0）です（表11）³¹. 正解の m の得点は勾配が負なので、上げる向きへ、ほかの6文字は正なので、下げる向きへ回されます. 4問の平均をとるので、最後に 4 で割ります.

表 11 mia の1問目（始まり → 正解 m）の7つの得点の勾配（学習前）

文字	確率	正解なら 1	確率－正解	÷4（問題の数）
a	12.7%	0	+0.127	+0.032
e	15.2%	0	+0.152	+0.038
i	15.3%	0	+0.153	+0.038
m	15.4%	1	−0.846	−0.211
n	14.4%	0	+0.144	+0.036
v	13.8%	0	+0.138	+0.034
終わり	13.3%	0	+0.133	+0.033

1回目の勾配：どの表が大きく効いたか 表ごとに勾配の大きさを比べると（図16），いちばん大きいのは lm_head（得点を出す表）です. 罰点にいちばん近い歯車だからです. Attention の表（attn_wq など）の勾配はほぼ 0 でした. 学習前は vv や wo の値が小さく、どの位

³⁰この方法を誤差逆伝播法（バックプロパゲーション, **backpropagation**）と呼びます. microgpt では Value という部品が、計算のたびに「何と何から、どんな計算で作ったか」と回転比（局所的な微分）を記録します（29–72行目）.

³¹確率と罰点（−ln）の組み合わせの微分がちょうど打ち消し合って、この形になります（付録A.7）. プログラムが計算した値と小数第3位まで一致しています.

置を見ても書き足す量がわずかなので、見る位置を変えても罰点がほとんど変わらないためと考えられます。268個のうち16個は勾配がちょうど0でした。miaに出てこない文字 e, n, v の wte の行（12個）と、使わなかった5番目の位置 p4 の wpe の行（4個）です。使わなかったつまみは罰点に影響しません。

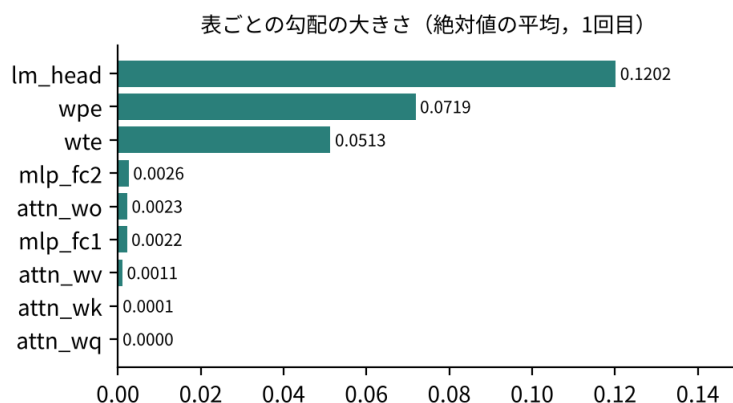


図 16 表ごとの勾配の大きさ（1回目）。

3.5 つまみを回す (Adam)

基本：勾配と反対の向きに、少しだけ回す 勾配が正なら減らし、負なら増やす。これを**勾配降下法** (gradient descent) と呼びます。一度に大きく回しすぎると、谷を飛び越えてしまいます。だから少しずつ回して、測り直します。1回に回す幅の目安を**学習率** (learning rate) と呼びます。microgpt では 0.01 から始めて、300回目に 0 になるよう少しずつ小さくします。最初は大大、谷に近づいたら小またで歩くのです。

Adam の2つの工夫 microgpt は、勾配降下法を改良した Adam (アダム) という方法を使います³²。

1. **勾配の平均を使う (勢い)** : 1回の練習は1つの名前だけなので、勾配は名前ごとにばらつきます。最近の勾配の平均を使って、ばらつきをならしめます。坂を転がるボールの勢いのように、小さなでこぼこでは向きを変えません。
2. **回す幅をそろえる** : つまみによって勾配の大きさは何桁もちがいます。それぞれの勾配の大きさで割り、どのつまみも同じくらいの幅で動くようにします。急な坂でも緩い坂でも、同じ歩幅で歩くようなものです。

1回目：どのつまみもほぼ 0.01 ずつ動いた 表12を見ると、勾配が正のつまみは減り、負のつまみは増えています。どれも罰点を下げる向きです。勾配が 0.46 でも 0.28 でも、動いた幅はほぼ同じ 0.01 です。Adam の「幅をそろえる」工夫で、1回目はどのつまみも学習率 (0.01) だけ動きます³³。勾配が 0 のつまみ (e の wte など16個) は動きません。

³²adaptive moment estimation (適応的モーメント推定) に由来する名前で、2014年に Kingma と Ba が発表しました。モーメントは統計の言葉で、1次のモーメントは平均、2次のモーメントは2乗の平均です。Adam は勾配の平均と勾配の2乗の平均を見積もって、回す向きと幅を決めます (付録A.8)。

³³1回目は勾配の平均も2乗の平均も、その1回の勾配だけから作られるので、「勾配 ÷ 勾配の大きさ」= ±1 になります。勾配がとても小さいつまみは、0 で割らないための小さな数の影響で、0.0099 くらいになります。

表 12 1回目のつまみの動き（冊子『microgpt を読む』5.3節と同じ）

つまみ	学習前の値	勾配	1回目の後	動いた幅
lm_head (i の行の3番目)	0.1090	+0.461	0.0990	-0.0100
lm_head (a の行の2番目)	0.0940	-0.438	0.1040	+0.0100
lm_head (m の行の4番目)	-0.0270	-0.390	-0.0170	+0.0100
wte (a の行の1番目)	0.0014	+0.282	-0.0086	-0.0100
wte (e の行の1番目)	-0.1026	0	-0.1026	0

1回目の後：mia は少しよくなった．ほかの名前は？ mia の4問は，どれも正解の確率が少し上がり，罰点は 1.953 から 1.889 に下がりました（表10右）．ほかの名前の罰点も調べると，mia と同じく「a で終わる」ava は 1.971 から 1.938 に下がりましたが，emma は 1.963 から 1.979 に上がり，anna は 2.001 から 2.002 とほぼ同じでした．

1つの名前に合わせた調整が，ほかの名前にもよいとは限りません．だから名前を替えながら何度もくり返し，全体として罰点が下がる場所を探します．

3.6 300回の練習

罰点の変化：ぎざぎざと，なめらか 図17の灰色は，その回に使った1つの名前の罰点です．名前ごとに難しさがちがうので，ぎざぎざします．緑は，そのときのつまみで4つの名前すべて（18問）を解いた平均です．1.97 から 0.347 まで，なめらかに下がりました．50回目あたりまでに急に下がり，あとはゆっくり 0.308 の線に近づきます³⁴．「何択で迷っているか」に直すと，7.2択から 1.41択になりました．

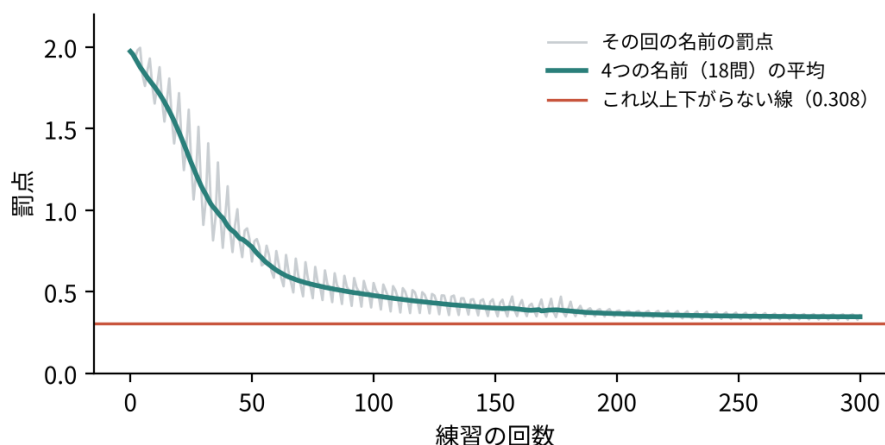


図 17 練習の回数と罰点.

AI は覚えやすいものから覚えていく 答えが一つに決まる場面の，正解の確率の変化を見ます（図18左）．名前の終わり（mia → 終わり）は早く覚えました．60回目で 86.7% です．m → i はなかなか覚えません．60回目で 21.0%，300回目でも 80.2% です．「名前の最後の a のあとは終わり」は4つの名前すべてに共通ですが，「m の次が i」は mia だけの癖です．たくさん名前に共通する規則ほど，練習の機会が多く，早く覚えています．

³⁴0.308 は，始まりの次と a の次で「正しく迷う」ことから来る限界です（1章，付録A.1）．

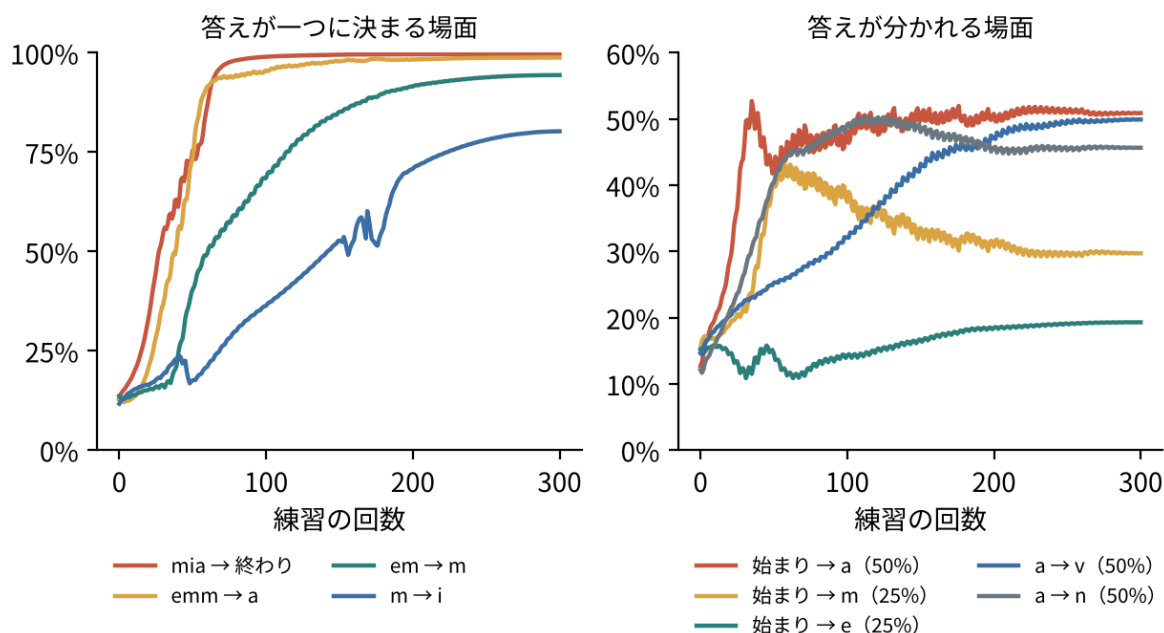


図 18 正解の確率の変化. 左：答えが一つに決まる場面. 右：答えが分かれる場面（カッコ内はデータの割合）.

迷う場面は、データの割合に落ち着く 答えが分かれる場面では（図18右），始まりの次の m は，60回目に行った 41.3% まで上がりすぎ，そのあと 30% 前後に戻りました．a の次は，はじめ n が先に伸び，v はあとから追いついて，300回目には v 49.9%，n 45.6% とほぼ半々になりました．どちらも，最後はデータの割合に近いところに落ち着きます．

i と v は，練習の中で同じ向きになった wte の i の行と v の行の向きのそろい方（cos）は，学習前は 0.64（偶然の向き）でした．10回目で 0.91，50回目には 1.00 になり，そのまま変わりませんでした（図19）．i（mia）と v（ava）は，どちらも「次は必ず a」という同じ役目です．同じ向きに置けば，同じ規則を使い回せます．だれも「i と v を似せよ」とは教えていません．罰点を下げる向きにつまみを回していたら，自然にそうになりました．

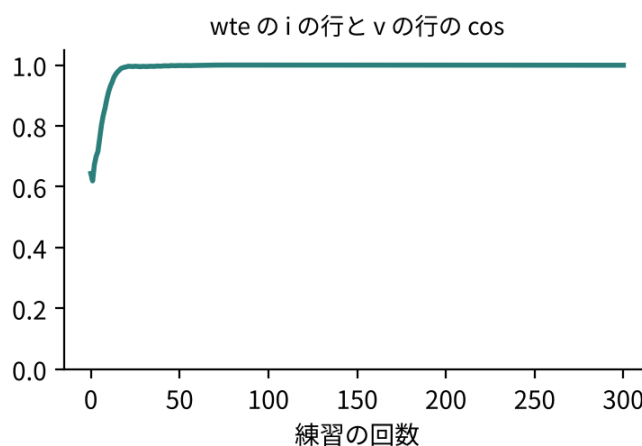


図 19 wte の i の行と v の行の向きのそろい方の変化.

300回で、つまみはどれだけ動いたか 表ごとに、学習前から学習後までの変化を比べると（図20）、最初はほとんど勾配がなかった Attention の表も、300回の間に大きく動きました。MLP の型紙（fc1 の16本の行）の長さは、学習前の 0.03～0.21 から、0.74～1.27 に育ちました。いちばん動きが小さかったのは wte と wpe でした³⁵。

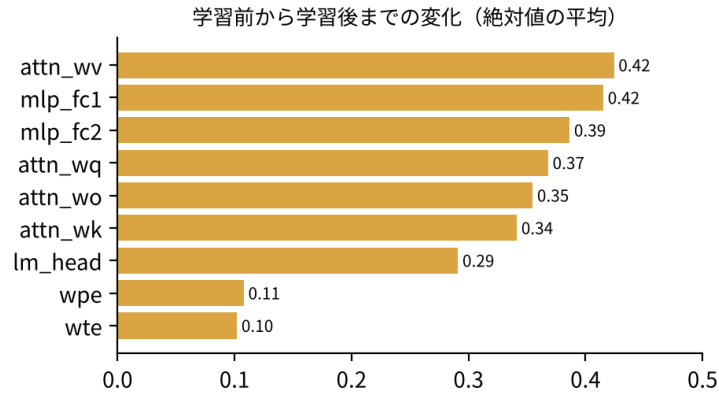


図 20 表ごとの、学習前から学習後までの変化（絶対値の平均）。

3.7 学習とは、最小二乗法と同じ考え方

理科の実験で、ばねにおもりをつるして伸びを測り、グラフに直線を引くとき、私たちは「データの点と直線のずれ」がなるべく小さくなるように、直線の傾きと切片を決めます（最小二乗法）。microgpt がしていることも本質的には同じです（表13）。

表 13 最小二乗法と microgpt の学習

	ばねの実験（最小二乗法）	microgpt
つまみ（パラメータ）	直線の傾きと切片（2個）	9つの表の数（268個）
外れ具合（損失）	点と直線のずれの2乗の合計	正解につけた確率の罰点の平均
学習	ずれが最小になる傾きと切片を求める	罰点が小さくなるつまみの位置を探す
求め方	式で一度に解ける	式では解けないので、勾配を見て少しずつ

AI の「学習」は、人間の学習とは意味がちがいます。損失が小さくなるようにつまみを調整すること、それ以上でも以下でもありません。それでも i と v が同じ向きになるような「整理」が、結果として生まれます。

3章のまとめ：外れ具合は「 $-\ln$ （正解の確率）」の平均で測ります。学習前は当てずっぽうと同じ 1.953 で、300回の練習で 0.347（限界 0.308）まで下がりました。勾配は、計算の記録を逆にたどって一度に求めます。得点の勾配は「確率 - 正解」というやさしい形になります。Adam で少しずつ回すと、AI は覚えやすいものから順に覚え、 i と v は自然に同じ向きになりました。

³⁵学習前の値は ± 0.16 くらいなので、平均 0.3～0.4 の変化は、最初の値よりずっと大きな動きです。

4 名前を作る

学習が終わったら、同じ予想の計算（2章）を使って新しい名前を作ります。これを**生成（generation）**と呼びます。つまみはもう回しません。新しいのは、確率をどう使って文字を選ぶかだけです。

4.1 名前を作る手順

1. 始まりの印（BOS）を読ませる。
2. 次の文字の確率を出す（2章の計算）。
3. 確率を当たりやすさにしたくじを引いて、1文字選ぶ。
4. 選んだのが終わり（BOS）なら完成。ちがえば、選んだ文字を付け足し、それを読ませて2へ戻る。

練習では正解の名前を読ませましたが、ここでは AI が自分で選んだ文字を読ませます。一度に読める長さは5なので、5文字目まで来ると、終わりが出なくてもそこで止まります³⁶。

4.2 確率をくじにする

学習後の AI は、始まりの次の文字に a 50.9%, m 29.7%, e 19.3% をつけます。これを、1000本のうち a の当たりが約509本、m が約297本、e が約193本入った箱から1本引くようなくじにします。a が出やすいのは確かですが、m や e が出ることもあります。引くたびに結果が変わるので、同じ AI から毎回ちがう名前が作られます。

くじの引き方 コンピュータは、0 以上 1 未満の数をでたらめに1つ選びます（乱数）。確率の帯を左から並べておき、選んだ数が入った区間の文字を当たりとします（図21）。区間の幅が確率に等しいので、確率の高い文字ほど当たりやすくなります。目をつぶってダーツを投げても、広い区画ほどよく当たるのと同じです³⁷。



図 21 確率の帯とくじ。0.62 が m の区間に入ったので、m が選ばれる。

なぜ「いちばん確率の高い文字」を選ばないのか いつも一番を選ぶと（貪欲法, greedy decoding），同じ AI からはいつも同じ名前しか出てきません。この AI なら、始まり → a（50.9%），a → v（49.9%）...と進み、毎回 ava だけになります。a の次は v と n がほとんど同じ確率なのに、必ず v を選んでしまいます。反対に、確率を無視して等しく選ぶと、学習前のようなでたらめな文字の列になります。確率を当たりやすさにしてくじを引くと、名前らしさを保ちながら、いろいろな名前を作れます。

³⁶プログラムでは 186–200 行目です。195 行目で確率を求め、196 行目の `random.choices` でくじを引きます。

³⁷乱数は、1 章で見た乱数の種 42 から計算で作るので、同じプログラムなら誰が動かしても同じ名前の列になります。

4.3 温度

確率に直す前（2章の softmax の前）に、7つの得点を**温度**（temperature）と呼ばれる数で割ります（図22）。温度が1より小さいと得点の差が広がり、確率の高い文字がもっと選ばれやすくなります。温度0.5では、始まりの次のaが50.9%から67.4%に増え、eは19.3%から9.7%に減ります。温度が1より大きいと差が縮まり、珍しい文字も選ばれやすくなります。microgptは温度0.5を使います³⁸。

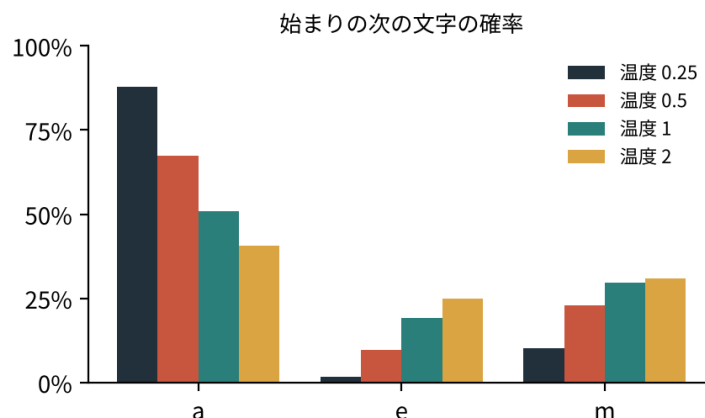


図 22 温度による、始まりの次の文字の確率のちがい。

温度は「慎重さ」と「冒険心」のつまみです。温度が低いと確率の高い文字ばかり選び、無難でよく見る名前が多くなります。0に近づけると、いつも一番を選ぶ貪欲法と同じになります。温度が高いと確率の低い文字もよく選び、変わった名前や崩れた名前が増えます。大きくすると、でたらめに近づきます。

温度は練習（学習）では使いません。名前を作るときに、使う人が選ぶ設定です。つまみを学習し直さなくても、温度だけ変えて作り直せます。ChatGPTやClaudeのような大きなAIでも同じで、開発者向けの使い方では温度を指定できます。「温度」という名前は物理学から来ています。得点を温度で割って確率にする式は、統計力学のボルツマン分布と同じ形です（付録A.9）。

4.4 1つの名前ができるまで

学習後のAIが最初に作った名前「ava」を、くじ1本ずつたどります（図23）。帯は温度0.5で調整した確率です。始まりの次では、くじ0.555がaの区間（0～0.674）に入り、aが選ばれました。aの次では、くじ0.932がvの区間に入り、vが選ばれました。avの次とavaの次は、確率がほぼ100%の1文字だけなので、どんな数を引いても同じ文字（a、終わり）になります。

³⁸温度1のときが、AIが計算したそのままの確率です。プログラムでは187行目で `temperature = 0.5` と決め、195行目で得点を温度で割っています。

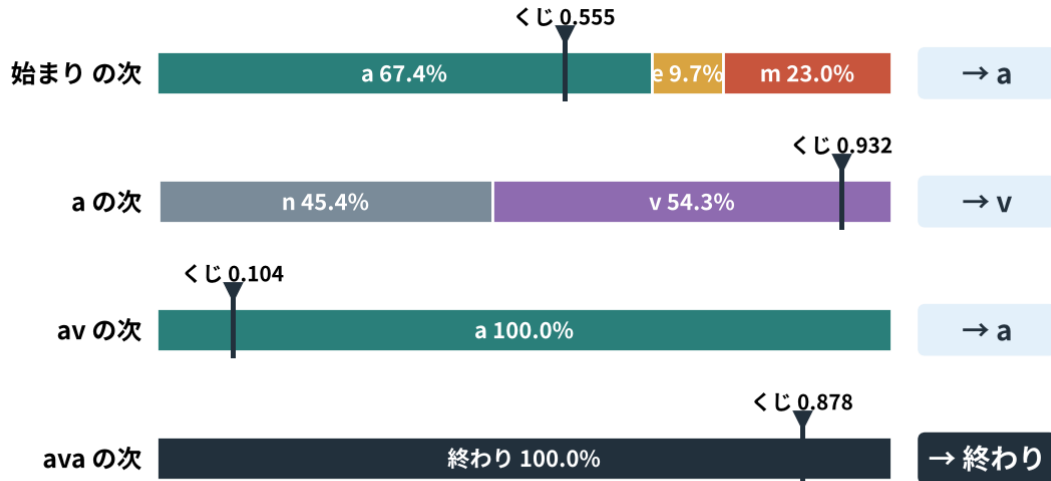


図 23 1つ目の名前「ava」ができるまで（温度 0.5）。黒い線がくじで選んだ数。

選んだ文字が、次に読む文字になる 練習のときは、正解の名前が最初から全部わかっていたので、すべての位置の予想をまとめて計算できました。名前を作るときは、次の文字がくじで決まるまで、その次を計算できません。だから1文字ずつ順にしか進めません。ChatGPT や Claude が答えを少しずつ表示していくのも同じ理由です。Attention が見に行く前の位置の見出し (k) と中身 (v) は、作った順にとっておいて使い回します³⁹。

4.5 作られた名前

学習後の AI に20個の名前を作らせると、次のようになりました（温度 0.5）⁴⁰。

ava, ava, anna, anna, mia, mia, anna, anna, ava, ava, ava, mia, anna, ava, ava, anna, mia, anna, emma, ava

ava が8個、anna が7個、mia が4個、emma が1個で、4つの名前以外は1つも出てきませんでした。データが少ないので、AI は名前を丸暗記した状態です。学んだ名前は4つとも同じ1回ずつなのに、ava と anna が多く、emma は1個だけです。なぜでしょうか。

emma が少ない理由：温度 0.5 が「e」をさらに減らす 名前の確率は、1文字ずつの確率のかけ算です。emma が選ばれるには、まず始まりの次で e が出る必要があります。始まりの次の e は、温度 1 で 19.3%、温度 0.5 では 9.7% しかありません。a は 50.9% から 67.4% に増えます。温度 0.5 は「多数派をもっと多く、少数派をもっと少なく」します。だから a で始まる ava と anna が増え、emma が減りました（表14）。

³⁹前の位置の k と v をとっておくしくみは、大きな AI で KV キャッシュ (KV cache) と呼ばれるものと同じです (121-122行目)。

⁴⁰元の microgpt.py の4か所の数字と input.txt だけを変えて動かした結果そのものです (冊子『microgpt を読む』5節と同じ)。

表 14 名前ができる確率（学習後）

名前	データ	温度 1 の確率	温度 0.5 の確率	20個中の個数
ava	25%	25.1%	36.6%	8
anna	25%	19.3%	30.0%	7
mia	25%	23.6%	22.6%	4
emma	25%	16.8%	9.6%	1
ほかの名前	0%	15.1%	1.2%	0

温度を変えて作る 温度だけを変えて20個ずつ作ると、次のようになりました⁴¹。下線はデータにない名前です。

- **温度 0.5**（データにない名前 0個）：上と同じ。
- **温度 1**（2個）：emma, mia, anna, ava, mia, mia, emma, anna, ava, ava, emma, anna, ava, emea
- **温度 2**（12個）：emeea, mnia, ava, ava, meia, anna, mnia, ema, m, anna, emna, anva, ava, iana, aia, mia, ava, av, mva, anva, emea, a など。

1000個ずつ作って数えると（表15、図24），温度 0.25 では4つの名前しか出ず，emma はとても少なくなります。温度 1 では4つがほぼ同じくらいになり，データにない名前が約15% 出てきます。温度 2 では半分以上がデータにない名前です（aia, mva, anva, emea, a など）。

表 15 1000個ずつ作ったときの個数（乱数の種 7）

温度	ava	anna	mia	emma	データにない名前
0.25	501	368	113	18	0
0.5	350	335	216	88	11
1	261	193	235	163	148
2	136	64	117	80	603

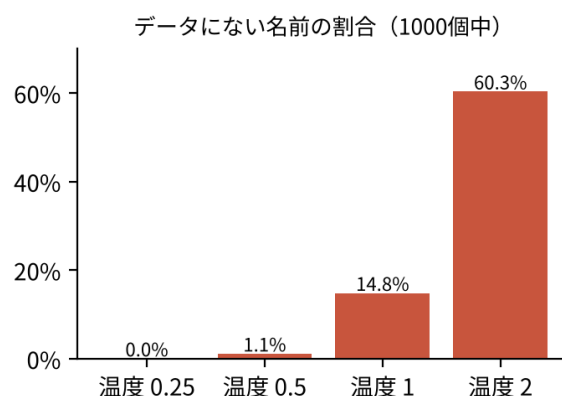


図 24 1000個のうち，データにない名前の割合。

⁴¹3つとも，学習を終えた直後の同じ乱数の続きを使い，温度だけを変えました。温度 0.5 の列は，元のプログラムの結果と同じです。

4.6 データにない名前は、覚えた部品の組み合わせ

データにない名前は、どのようにしてできたのでしょうか。各位置で選ばれた文字の確率（温度 1）を並べると、次のようになります。

名前	1文字ずつの確率
ava	始まり → a 50.9%, a → v 49.9%, av → a 99.2%, ava → 終わり 99.8%
aia	始まり → a 50.9%, a → i 3.5% , ai → a 99.2%, aia → 終わり 99.8%
anva	始まり → a 50.9%, a → n 45.6%, an → v 10.8% , anv → a 98.8%, anva → 終わり 99.8%

aia は、始まりの次に a を選び、次に珍しい i（3.5%）を引いた名前です。そのあとは「i の次は a」「名前の最後の a のあとは終わり」という、mia で覚えた規則がそのまま使われました。anva も、an の次に珍しい v（10.8%）を引いたあと、「v の次は a」（ava で覚えた）で続いています。新しい名前は、でたらめではなく、覚えた部品（規則）をつなぎかえたものです。

4.7 練習の前と途中で作った名前

学習前のつまみで作った名前は、でたらめな文字の列です（温度 0.5，乱数の種 7）。

ianam, iamai, aaiva, en, mi, aviea, ivenn, imaae, nminm, ivnim, m, ne, eivem, anvnv, innnm, v, mnann, (空), veina, meaav

7文字にほぼ同じ確率をつけるので、「終わり」もほかの文字と同じくらいしか選ばれず、1文字で終わったり（m, v）、何も作らずに終わったり（空）します。5文字のものは、一度に読める長さ（5）で止められた名前です。

練習の途中で10個ずつ作ると、名前は表16のように変わりました⁴²。覚えやすい規則から順に覚えるので（3章）、作る名前も「a ばかり」→「終わり方が名前らしい」→「4つの名前」と変わっていきました。

表 16 練習の途中で作った名前（10個ずつ）

練習	作った名前	3章で見たこと
学習前	ianam, iamai, aaiva, en, mi, aviea, ivenn, imaae, nminm, ivnim	7文字にほぼ同じ確率
25回	aeaa, aeaa, aea, aia, an, avaaa, ava, eeaaa, annnv, aim	まず「a が多い」ことを覚える
50回	aiaa, aiaa, aia, ann, mnva, aia, mia, ania, mnn, anva	終わりの規則を覚えつつある
100回	anna, aia, anna, anna, ava, anna, anna, aia, anna, mnna	emm → a, em → m も覚えた
300回	anna, anna, anna, ava, ava, anna, ava, anna, anna, mia	m → i も 80% まで覚えた

⁴²温度 0.5，どの回も乱数の種 7 で作りました。学習の乱数には影響させていません（冊子『microgpt を読む』付録Dと同じ方法）。

4.8 データを増やすと

元の設定（3万2千個の名前，1000回の練習，温度 0.5）で作った20個の名前は，次のとおりです（冊子『microgpt を読む』7.2節）。

kamon, ann, karai, jaire, vialan, karia, yeran, anna, areli, kaina, konna, keylen, liole, alerin,
earan, lenne, kana, lara, alela, anton

ann, anna, lara, anton のように実在する名前もありますが，kamon や vialan のように，データにはないが名前らしいものが多く含まれます。3万2千個は覚えきれないので，AI は「名前によく出る文字の並び方」の統計を学んでいます。4つの名前では丸暗記，3万2千個では部品の統計。同じプログラムでも，データの量で AI の「覚え方」が変わります。

「新しい名前」と，大きな AI の作り話 microgpt のプログラムは，作った名前を表示するとき「new, hallucinated names」（新しい，幻の名前）と書きます。hallucination（幻覚）は，大きな AI が，事実でないことをもっともらしく作ってしまうことを指す言葉としても使われます。しくみは同じです。大きな AI も，覚えた言葉の並び方の確率に従って，次のトークンをくじで選んでいるだけです。並び方としてもっともらしければ，事実かどうかに関係なく作られます⁴³。kamon は「名前らしい」けれど，実在の名前とは限りません。大きな AI の答えも，「文章としてもっともらしい」とことと「正しい」ことは別です。確かめる習慣が大切です。

4章のまとめ：名前は，確率のくじを1本ずつ引いて作ります。始まりから次の文字の確率を出してくじを引き，選んだ文字を読ませ，終わりが出たら完成です。温度でくじの当たりやすさを調整します。温度 0.5 は多数派をもっと多くするので，a で始まる ava と anna が多く，emma は少なくなりました。新しい名前は，覚えた部品の組み合わせです。4つの名前では丸暗記に近く，データを増やすと「名前らしさ」の統計から新しい名前が作られます。

⁴³大きな AI では，事実に合う答えを選ぶように追加の学習をしたり，調べた資料をもとに答えさせたりして，作り話を減らす工夫がされています（「LLMの基礎3 対話モデルへ」参照）。

5 本物の AI と比べる

microgpt のプログラムの冒頭には「このファイルがアルゴリズムのすべてで、それ以外は効率化にすぎない」という趣旨のことが書かれていました。この章では、この主張を確かめます。

5.1 大きさを比べる

表17は、小さい例、元の microgpt, GPT-2 (最小のもの), GPT-3 を並べたものです⁴⁴。GPT-2 は 2019年, GPT-3 は 2020年に OpenAI が発表した言語モデルで, ChatGPT のもとになった GPT シリーズの初期のものです。microgpt のモデルは GPT-2 とほぼ同じ構造です。ちがいは主に、この表の数の大きさです。

表 17 4つのモデルの大きさ

	小さい例	microgpt	GPT-2 (最小)	GPT-3
黒板の幅 (1トークンあたりの数)	4	16	768	12,288
層の数 (Attention + MLP の組)	1	1	12	96
Attention のヘッドの数	1	4	12	96
一度に読める長さ	5	16	1,024	2,048
語彙 (トークンの種類)	7	27	50,257	50,257
つまみの数	268	4,192	約1.24億	約1,750億

つまみの数を「1個 = 1 mm」で並べると、小さい例 (268個) は 26.8 cm でものさしくらい、microgpt (4,192個) は約 4.2 m で教室の幅くらい、GPT-2 最小 (約1.24億個) は約 124 km でフルマラソン約3回分、GPT-3 (約1,750億個) は約 17万5千 km で地球を約4.4周する長さです⁴⁵。GPT-3 のつまみは、microgpt の約4,000万倍、小さい例の約6億5千万倍です (図25左)。

⁴⁴GPT-2 と GPT-3 の数は、冊子『microgpt を読む』8.1節の表と同じです。ChatGPT や Claude のような最近の大きな AI は、構造や大きさがくわしく公表されていないものが多いので、ここでは公表されている GPT-2 と GPT-3 で比べます。

⁴⁵地球一周は約4万 km、フルマラソンは 42.195 km として計算しました。

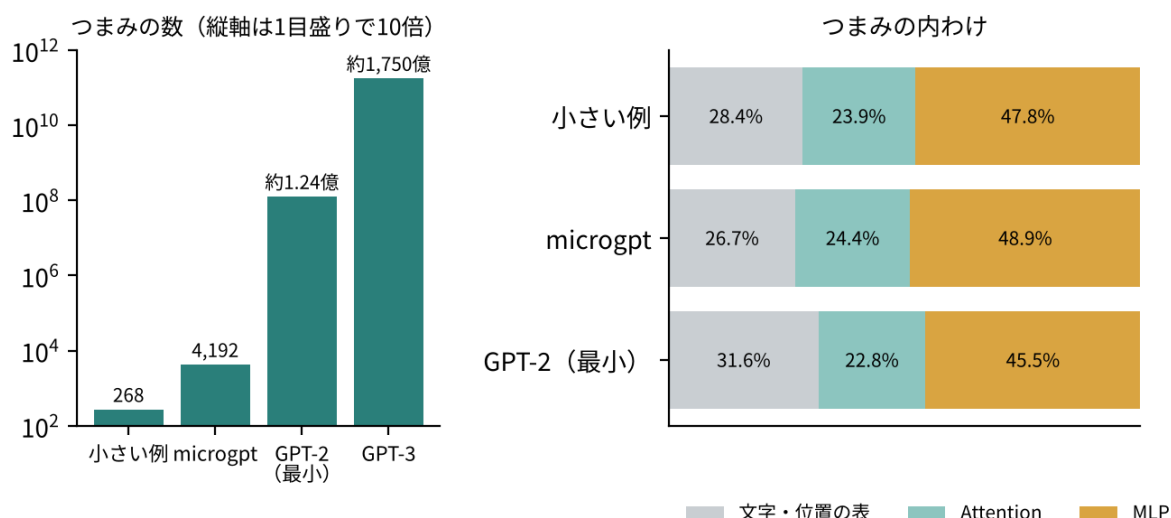


図 25 左：つまみの数（縦軸は1目盛りで10倍）．右：つまみの内わけ．

つまみの配分は、大きさが変わってもよく似ている つまみを「文字・位置の表 (wte, wpe, lm_head)」「Attention」「MLP」に分けると (図25右), どのモデルでも MLP がほぼ半分 (48%, 49%, 46%), Attention が約4分の1です⁴⁶. 2章で見たように, 「こういうときは次にこれ」という規則の多くを持つ MLP に, いちばん多くのつまみが使われています. 大きさは約46万倍ちがっても, 配分の考え方は同じです.

層を重ねる microgpt は Attention と MLP の組を1回だけ通します (1層). GPT-2 最小は12回, GPT-3 は96回通します. 黒板は1枚のまま, Attention と MLP の組が何回も書き足していきます. 前の層が書き足した結果を次の層が読めるので, 「見つけたものを組み合わせ, さらに見つける」ことができます. 組み立て工場のラインのように, 1つの工程では簡単な作業しかなくても, 工程を重ねると複雑な製品ができます.

ヘッドと, 一度に読める長さ ヘッドは, Attention を何組に分けて行うかの数です. 黒板の幅を分けて, 組ごとに別の位置を見に行きます. 複数のヘッドは, 同じ文を「文法の係」「意味の係」のように分担して読むようなものです (ただし, 実際の分担は学習で決まり, 人間にきれいに分かるとは限りません). 一度に読める長さは, Attention が見られる位置の数の上限です. どの位置も自分より前の位置を全部見るので, 長さが2倍になると, 見る組み合わせの数は約4倍になります (表18, 付録A.11).

⁴⁶GPT-2 の数は, 公表されている構造 (黒板の幅 768, 12層, 語彙 50,257, 長さ 1,024) から数えたもので, 合計 124,439,808 個です (付録A.10). GPT-2 の Attention と MLP の数には定数項 (バイアス) も含めました. 正規化のつまみ (0.03%) は図から省いています. GPT-2 は lm_head を wte と共有しています.

表 18 一度に読める長さと、Attention が見る組み合わせの数

一度に読める長さ n	見る組み合わせの数 $n(n+1)/2$
5 (小さい例)	15
16 (microgpt)	136
1,024 (GPT-2)	524,800
2,048 (GPT-3)	2,098,176

語彙 microgpt の1つのトークンは1文字ですが、GPT-2 と GPT-3 は、データの中でよく出る文字の組（単語や単語の一部）を1つのトークンとしてまとめます⁴⁷。よく出る単語は1トークン、めったに出ない単語はいくつかのかけらに分かれます。語彙が大きくなると wte と lm_head の表が大きくなり、GPT-2 では 50,257 行 × 768 列で、つまみの約3割がこの表です。AI が知っているのは番号だけ、という点は同じです。

5.2 データと練習の量

表 19 練習に使うデータの量

	小さい例	microgpt	GPT-3
データ	名前 4個	名前 32,033個	大量の文章
練習問題の数（トークンの数）	18	228,146	約 3,000億
練習の回数	300	1,000	—
1回の練習で使うデータ	名前 1個	名前 1個	多くの文書をまとめて

元の microgpt のデータは 32,033個の名前、196,113文字で、練習問題にすると 228,146問です（表19）⁴⁸。ところが元の microgpt は、1000回の練習で名前を1000個しか使いません。データ全体の約3%です。それでも、名前らしい新しい名前を作れるようになりました（4章）。大きな AI は、1回の練習で多くの文書をまとめて読み、その平均の勾配でつまみを回します。

データを増やすと、覚え方が変わる 4つの名前ではほぼ丸暗記でした。データにない名前は、温度を上げたときに部品をつなぎかえて出てくるだけでした。名前が多すぎて覚えきれないと、AI は「名前によく出る文字の並び方」という統計を学ぶしかなくなります。大きな AI でも、つまみの数とデータの量を増やすほど、初めて見る文章での罰点（損失）が下がる傾向が知られています（スケーリング則⁴⁹）。

⁴⁷よく出る文字の組をまとめる方法の一つに BPE (byte pair encoding) があります。くわしくは「LLMの基礎1 トークンと言語モデル」を参照してください。

⁴⁸196,113文字に、名前ごとの「終わり」を1つずつ足した数です。GPT-3 の数は発表論文 (Brown ほか, 2020) によります。

⁴⁹「LLMの基礎2 事前学習とスケーリング則」を参照してください。

5.3 計算のしかた

microgpt は、数を1つずつ Value という部品に入れて、足し算や掛け算を1回ずつ記録しながら計算します。Python の標準機能だけで動き、計算の中身が全部見えますが、その代わり遅く、小さい例で約2秒、元の設定で2〜3分かかります。大きな AI は、数の表（行列）をまとめて掛け算します。GPU という、同じ種類の計算を大量に同時に行える装置を、たくさん並べて使います⁵⁰。計算の式は同じで、計算のしかたが速いのです。microgpt は1人が電卓で1つずつ計算するようなもの、大きな AI は体育館いっぱいの人が同じ形の計算を一斉に行うようなものです。答えは同じでも、かかる時間がまるでちがいます。

たとえば「表と黒板の掛け算」は、microgpt では行の数だけ積の和をくり返しますが、行列でまとめれば $y = Wx$ の1回で書けます。Attention の得点も、前の位置ごとに q と k の積の和を求める代わりに、全位置の q と k を行列 Q, K にまとめて $QK^T/\sqrt{d}$ を一度に計算します。練習のときは答えの文章が最初から全部わかっているので、すべての位置の予想を一度にまとめて計算できます。名前や文章を作るときに1文字（1トークン）ずつ順にしか進めないのは、大きな AI も同じです。

5.4 構造の小さなちがい

プログラムの93行目の注釈には、GPT-2 とのちがいが3点だけ書かれています（表20）。どれも「計算を簡単にする」ための変更です。このうち rmsnorm とバイアスなしは、最近の大きな AI（LLaMA など）でも使われている選び方です。黒板（残差の流れ）、Attention、MLP、lm_head、softmax という部品の組み立ては GPT-2 と同じです。

表 20 GPT-2 とのちがい

	GPT-2	microgpt	ちがい
大きさをそろえる計算	layernorm	rmsnorm	layernorm は平均を引いてからそろえる。rmsnorm は平均を引かない（簡単）
定数項（バイアス）	あり	なし	掛け算のあとに足す一定の数。microgpt は省いた
MLP の折り曲げ	GELU	ReLU	GELU は 0 の近くでなめらかに曲がる。ReLU は 0 でカクッと曲がる

学習後の調整：対話のできる AI にする microgpt が行うのは、大量のデータで「次のトークン」を当てる練習（事前学習）だけです。ChatGPT や Claude のような対話型の AI は、事前学習のあとに、質問と答えの例で指示に従うよう追加の学習（指示調整）をし、よい答えと悪い答えの人間の評価から好まれる答え方を学びます（RLHF など）⁵¹。どの段階でも、つまみを罰点が下がる向きに回すという原理は同じです。何を「よい答え」とするか（損失の決め方）がちがいます。

⁵⁰大きな AI の開発では、行列の計算を GPU で速く行うためのライブラリ（PyTorch など）を使います。microgpt はそれを使わず、すべてを自分で書いています。

⁵¹RLHF は reinforcement learning from human feedback（人間の評価からの強化学習）の略です。くわしくは「LLMの基礎3 対話モデルへ」を参照してください。

5.5 同じところ

骨組みは同じです（図26）。予想は、トークンを番号にし、表を引いて最初のメモを作り、Attention と MLP を層の数だけくり返し、得点を出して softmax で確率に直します。学習は、罰点（ $-\ln$ 正解の確率）を測り、勾配を逆にたどって求め、つまみを回します。生成は、温度で調整した確率でくじを引き、選んだトークンを付け足して読むことをくり返します。



図 26 microgpt と大きな AI に共通の骨組み.

1〜4章で見たことは、大きな AI ではどうなっているでしょうか（表21）⁵².

表 21 microgpt で見たことと、大きな AI

microgpt で見たこと	大きな AI では
AI が知っているのは番号だけ（1章）	同じ。トークンの番号だけを扱う
正しく迷う。罰点は 0 にならない（1章）	同じ。文章の続きは一つに決まらないことが多い
同じ役目の文字は同じ向きに置かれる（i と v）	似た使われ方の単語は近い向きになることが知られている
黒板に書き足していく（2章）	同じしくみ。黒板の幅は 768〜12,288 と広い
覚えやすいものから覚える（3章）	よく出るパターンほど早く学ばれる傾向がある
温度とくじ（4章）	同じ。使う人が温度を選べる
もっともらしい新しい名前（4章）	もっともらしい作り話（幻覚）

ちがうところ：中を読むのがずっと難しい 小さい例の 268 個のつまみは、冊子『小さなAIの頭の中をのぞく』で、ほぼ全部の役目を読み解けました（i と v, 16個の検出器など）。それでも、1つ1つのマスの値には意味がなく、「規則の寄せ集め」として覚えていることがわかっただけでした。1,750億個のつまみになると、中で何が起きているのかを人間が読み解くのはとても難しくなります。これを調べる研究（解釈可能性の研究, interpretability）が、いまでも盛んに行われています。大きな AI が計算問題を、きれいな手順ではなく「たくさんの小さ

⁵²右の列は、大きな AI について一般に知られている傾向を簡単にまとめたものです。すべての場合に当てはまるとは限りません。

な規則の寄せ集め」で解いていることを示した研究もあります⁵³。268個の小さな AI で見えた姿と、よく似ています。

5章のまとめ：大きさは桁ちがいで、つまみは GPT-3 で約1,750億個（小さい例の約6億5千万倍）です。ちがうのは主に計算のしかた（行列でまとめて GPU で一斉に計算する）と、構造の小さな3点（rmsnorm, バイアスなし, ReLU）です。骨組み（番号 → メモ → Attention と MLP → 確率）と学習の原理（罰点を勾配で下げる）は同じです。

最初の問いへの答え：「このファイルがアルゴリズムのすべてで、それ以外は効率化にすぎない」という microgpt の主張は、計算の骨組みについてはそのとおりでした。ただし、対話のための追加の学習や、大きさから生まれる力は、200行の中には入っていません。

おわりに：microgpt を解剖して見えたこと

1. **全体像と準備：**AI は「次を当てる」をくり返します。名前は番号の列になり、18問の練習問題になります。答えが分かれる場面では、正しく迷うのが最善です。
2. **1文字を読む：**最初のメモ → Attention → MLP → 得点 → 確率。得点は3つの書き込みの足し算で、いちばん効くのは MLP です。
3. **外れ具合と学習：**罰点を $-\ln$ で測り、勾配を逆にたどって求め、Adam で回します。覚えやすいものから覚え、 i と v は自然に同じ向きになりました。
4. **名前を作る：**確率のくじを1本ずつ引きます。温度で顔ぶれが変わります。新しい名前は、覚えた部品の組み合わせです。
5. **本物の AI と比べる：**大きさと計算のしかたはちがいますが、骨組みと学習の原理は同じです。

次に読むものとして、microgpt の200行を元の設定（3万2千個の名前）も含めて順に読む冊子『microgpt を読む』、同じ4つの名前で学習後の268個のつまみの意味を読み解く冊子『小さなAIの頭の中をのぞく』、トークンと言語モデル・事前学習とスケーリング則・対話モデルを扱う「LLMの基礎」1～3、ニューラルネットワークから Transformer までを扱う「深層学習」1～4があります。

⁵³Nikankin ほか「Arithmetic Without Algorithms: Language Models Solve Math With a Bag of Heuristics」。

付録A 数式

本文の計算を数式で書きます。理系の大学1~2年生程度を想定しています。記号は次のとおりです。語彙の大きさ $V = 7$ ，黒板の幅 $d = 4$ (n_embd)，一度に読める長さ $B = 5$ ($block_size$)。表は E ($wte, V \times d$)， P ($wpe, B \times d$)， W_Q, W_K, W_V, W_O ($d \times d$)， W_1 ($fc1, 4d \times d$)， W_2 ($fc2, d \times 4d$)， U ($lm_head, V \times d$) です。1つの名前を番号の列 $x_0, x_1, \dots, x_n$ ($x_0 = x_n = 6$, BOS) で表します。

A.1 答えが分かれる場面と、損失の下限 (1章)

AI が求めるのは、読んだところ $x_0 \dots x_{t-1}$ を条件とした次の文字の確率 $P(x_t | x_0, \dots, x_{t-1})$ です。名前全体の確率は、確率の乗法定理により

$$P(\text{名前}) = P(x_1 | x_0) P(x_2 | x_0, x_1) \cdots P(x_n | x_0, \dots, x_{n-1}) \quad (1)$$

です。データの割合で計算すると $P(\text{emma}) = \frac{1}{4} \times 1 \times 1 \times 1 \times 1 = \frac{1}{4}$ ， $P(\text{ava}) = \frac{2}{4} \times \frac{1}{2} \times 1 \times 1 = \frac{1}{4}$ で、4つとも $\frac{1}{4}$ になります。

データの割合どおりに答えたときの18問の罰点の合計は、始まりの4問で $-\ln \frac{2}{4}$ が2回と $-\ln \frac{1}{4}$ が2回，a の次の2問で $-\ln \frac{1}{2}$ が2回，ほかの12問は $-\ln 1 = 0$ なので

$$2(0.693) + 2(1.386) + 2(0.693) = 5.545, \quad 5.545 \div 18 = 0.308 \quad (2)$$

です。これはデータに残るあいまいさ（条件付きエントロピー）で、どんな予想をしても罰点の平均はこれより小さくなりません（ギブスの不等式）。 $e^{\text{罰点の平均}}$ をパープレキシティ（perplexity）と呼び、「平均して何択で迷っているか」を表します。限界は $e^{0.308} = 1.36$ 択，学習前は $e^{1.973} = 7.2$ 択，学習後は $e^{0.347} = 1.41$ 択です。

A.2 最初のメモ，rmsnorm，cos (2章)

$$\text{rms}(z) = \frac{z}{\sqrt{\frac{1}{d} \sum_i z_i^2 + \varepsilon}}, \quad \varepsilon = 10^{-5}, \quad \mathbf{x}_t^{(0)} = \text{rms}(E_{x_t} + P_t) \quad (3)$$

ε を無視すれば， $\text{rms}(z)$ の長さは $\sqrt{d} = 2$ です。microgpt は Attention と MLP の直前にも rmsnorm をかけますが、最初のメモはすでに長さ 2 なので、Attention の直前の rmsnorm ではほとんど変わりません。2つのベクトルの向きのそろい方は

$$\cos \theta = \frac{\mathbf{a} \cdot \mathbf{b}}{\|\mathbf{a}\| \|\mathbf{b}\|}, \quad \mathbf{a} \cdot \mathbf{b} = \sum_i a_i b_i \quad (4)$$

です。p1 の e と p2 の m は $\cos \theta = 0.98$ ，p2 の m と p3 の m は -0.41 でした。

A.3 Attention と softmax (2章)

$\hat{\mathbf{x}}_t = \text{rms}(\mathbf{x}_t^{(0)})$ として

$$\mathbf{q} = W_Q \hat{\mathbf{x}}_t, \quad \mathbf{k}_s = W_K \hat{\mathbf{x}}_s, \quad \mathbf{v}_s = W_V \hat{\mathbf{x}}_s, \quad s_s = \frac{\mathbf{q} \cdot \mathbf{k}_s}{\sqrt{d}}, \quad a_s = \frac{e^{s_s}}{\sum_{s'=0}^t e^{s_{s'}}} \quad (5)$$

$$\mathbf{x}_t^{(1)} = \mathbf{x}_t^{(0)} + W_O \sum_{s=0}^t a_s \mathbf{v}_s \quad (6)$$

$s \leq t$ なので、先の位置は見られません（因果的な Attention）。p3 の例では、得点 $(+2.69, -0.11, +0.00, +1.00)$ ，割合 $(0.762, 0.046, 0.052, 0.140)$ です。元の microgpt は $n_head = 4$ で、16個の数を4つずつに分けて、4組の Attention を別々に行います（マルチヘッド）。

A.4 MLP (2章)

$$\mathbf{u} = W_1 \text{rms}(\mathbf{x}_t^{(1)}), \quad h_i = \text{ReLU}(u_i) = \max(0, u_i), \quad \mathbf{x}_t^{(2)} = \mathbf{x}_t^{(1)} + W_2 \mathbf{h} \quad (7)$$

u_i は fc1 の i 行目 (型紙 κ_i) と, 大きさをそろえた黒板 $\hat{\mathbf{x}}$ の内積です. $|\hat{\mathbf{x}}| = 2$ なので $u_i = 2|\kappa_i| \cos \theta_i$ で, 型紙と同じ向きに近いほど大きく, 90° を超えると負 (黙る) になります. 書き足す量 $W_2 \mathbf{h} = \sum_i h_i \mathbf{w}_i$ ($\mathbf{w}_i$ は fc2 の i 列目) は, 点灯した検出器の「足す内容」を点灯の強さで重みづけした和です. ReLU がなければ $W_2 W_1$ は1つの 4×4 行列と同じになり, 16個に広げる意味がありません.

A.5 得点の3つへの分解 (2章)

$\mathbf{x}_t^{(2)} = \mathbf{x}_t^{(0)} + \mathbf{A} + \mathbf{M}$ ($\mathbf{A}$: Attention が足した分, $\mathbf{M}$: MLP が足した分) なので

$$\ell = U \mathbf{x}_t^{(2)} = U \mathbf{x}_t^{(0)} + U \mathbf{A} + U \mathbf{M}, \quad p_j = \frac{e^{\ell_j}}{\sum_k e^{\ell_k}} \quad (8)$$

が正確に成り立ちます. lm_head の前に rmsnorm がいないことが条件です. softmax は, 7つの得点すべてに同じ数を足しても変わらない (分子も分母も同じ倍率になる) ので, 意味があるのは得点の差です. $p_a/p_e = e^{\ell_a - \ell_e}$ で, p3 では $e^{4.57} \approx 96$ でした. プログラムでは, 数が大きくなりすぎないように, いちばん大きい得点を引いてから e の何乗かを求めています (97-101行目).

A.6 損失 (交差エントロピー) (3章)

$$\text{Loss} = \frac{1}{n} \sum_{t=1}^n (-\ln p_t(\text{正解})) \quad (9)$$

mia ($n = 4$) の1回目は $(1.869 + 2.151 + 1.764 + 2.029)/4 = 1.953$ です. 7文字に同じ確率 $1/7$ をつけると $\text{Loss} = \ln 7 \approx 1.946$, 元の microgpt (27文字) なら $\ln 27 \approx 3.30$ です. 正解の文字を 1, ほかを 0 とした分布 $\mathbf{y}$ と, AI の確率 $\mathbf{p}$ の交差エントロピー $H(\mathbf{y}, \mathbf{p}) = -\sum_j y_j \ln p_j$ が, ちょうど $-\ln p(\text{正解})$ になります.

A.7 勾配, 連鎖律, 得点の勾配 (3章)

勾配は, つまみ w を小さな数 Δ だけ動かしたときの罰点の変化の割合です.

$$\frac{\partial \text{Loss}}{\partial w} \approx \frac{\text{Loss}(w + \Delta) - \text{Loss}(w)}{\Delta} \quad (10)$$

本文の例 ($\Delta = 0.001$) では $(1.953936 - 1.953475)/0.001 = 0.461$ で, 自動微分の値 0.4608 と一致しました. Loss が y を通して w に依存するとき $\frac{\partial \text{Loss}}{\partial w} = \frac{\partial \text{Loss}}{\partial y} \frac{\partial y}{\partial w}$ (連鎖律) で, 道が複数あるときは各道の積を足します. $f = ab + a$ なら $\partial f / \partial a = b + 1 = 4$, $\partial f / \partial b = a = 2$ です. 足し算 $z = x + y$ の局所的な微分は $\partial z / \partial x = 1$, 掛け算 $z = xy$ は $\partial z / \partial x = y$ (相手の値) で, microgpt の Value はこれを計算のたびに記録します.

得点 z から確率と損失を作ると, $\ell = -\ln p_c = -z_c + \ln \sum_k e^{z_k}$ (c : 正解) なので

$$\frac{\partial \ell}{\partial z_j} = -y_j + \frac{e^{z_j}}{\sum_k e^{z_k}} = p_j - y_j \quad (11)$$

になります. $\ln \sum e^z$ を z_j で微分すると, ちょうど softmax の p_j になるためです. 勾配の和は $\sum_j (p_j - y_j) = 1 - 1 = 0$ で, 正解の得点を上げた分だけほかを下げる形です.

A.8 Adam (3章)

勾配を g_t として ($m_0 = v_0 = 0$)

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t, \quad v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2 \quad (12)$$

$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t}, \quad \hat{v}_t = \frac{v_t}{1 - \beta_2^t} \quad (13)$$

$$w \leftarrow w - \eta_t \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon}, \quad \eta_t = 0.01 \left(1 - \frac{t-1}{300}\right) \quad (14)$$

$\beta_1 = 0.85$, $\beta_2 = 0.99$, $\epsilon = 10^{-8}$ です (174-182行目) . 1回目 ($t = 1$) は $\hat{m}_1 = g_1$, $\hat{v}_1 = g_1^2$ なので $\hat{m}_1/\sqrt{\hat{v}_1} = \pm 1$ となり, すべてのつまみが $\eta = 0.01$ だけ動きます ($g = 0$ なら動かない) . $\beta_1 = 0.85$ は, およそ直近 $1/(1 - 0.85) \approx 7$ 回分の勾配の平均にあたります.

A.9 くじ, 温度, 名前の確率 (4章)

累積の確率 $C_j = p_1 + \dots + p_j$ ($C_7 = 1$) を作り, $0 \leq r < 1$ の一様乱数 r に対して $C_{j-1} \leq r < C_j$ となる文字 j を選びます. r がこの区間に入る確率は区間の幅 p_j に等しいので, 文字 j は確率 p_j で選ばれます. 温度 T のときの確率は

$$p_j(T) = \frac{e^{z_j/T}}{\sum_k e^{z_k/T}} \quad (15)$$

です. 統計力学のボルツマン分布 $p_j = e^{-E_j/kT} / \sum_i e^{-E_i/kT}$ (E : エネルギー, k : ボルツマン定数) で, 得点 z をエネルギーに負の符号をつけたものと見ると, 同じ形になります. 2つの文字の確率の比は $p_j/p_k = e^{(z_j - z_k)/T}$ で, $T = 0.5$ では比が2乗になります. $T \rightarrow 0$ ではいちばん得点の高い文字だけが選ばれ (貪欲法), $T \rightarrow \infty$ ではすべての文字が $1/7$ に近づきます.

名前の確率は各位置の (温度で調整した) 確率の積で, N 個作ったときの個数の見込み (期待値) は $N \times P(\text{名前})$ です. 温度 1 で $P(\text{aia}) = 0.509 \times 0.035 \times 0.992 \times 0.998 = 0.0175$ なので, 1000個で約17個の見込み (実際は22個) . 温度 0.5 で $P(\text{emma}) = 0.096$ なので, 20個では見込み約1.9個 (実際は1個) , 1000個では約96個 (実際は88個) です.

A.10 つまみの数 (5章)

1層の中身は, Attention の4つの表で $4d^2$, MLP の2つの表で $4d \times d + d \times 4d = 8d^2$, 合わせて $12d^2$ です.

$$\text{microgpt: } 2Vd + Bd + 12d^2L \quad (16)$$

$$\text{GPT-2, GPT-3: } Vd + Bd + L(12d^2 + 13d) + 2d \quad (17)$$

GPT-2 と GPT-3 では `lm_head` を `wte` と共有し, バイアスと正規化のつまみ ($13dL + 2d$) を含みます. 小さい例 ($V = 7, d = 4, B = 5, L = 1$) は $56 + 20 + 192 = 268$, `microgpt` (27, 16, 16, 1) は 4,192, GPT-2 最小 (50, 257, 768, 1, 024, 12) は 124,439,808, GPT-3 (50, 257, 12, 288, 2, 048, 96) は約 1,746 億 (約1,750億) で, 大部分は $12d^2L$ の部分です.

A.11 Attention の計算の量 (5章)

長さ n の文で, 位置 t は t 個の位置を見るので, 組み合わせの数は $1 + 2 + \dots + n = n(n+1)/2$ です. 1組ごとに q と k の内積 (d 回の掛け算) が必要なので, 計算の量はおよそ n^2d に比例

します。練習では全位置の q と k を行列 Q, K にまとめて $QK^T/\sqrt{d}$ を一度に計算し、先の位置を見ないように右上の部分の部分を隠してから softmax をとります。名前や文章を作るときは、前の位置の k と v をとっておく (KV キャッシュ) ので、新しい1位置ぶんの計算だけで済みます。

A.12 layernorm と GELU (5章)

$$\text{layernorm}(x) = \gamma \odot \frac{x - \mu}{\sqrt{\sigma^2 + \varepsilon}} + \beta, \quad \text{GELU}(u) = u \Phi(u) \quad (18)$$

μ は平均, σ^2 は分散, γ, β は学習するつまみ, Φ は標準正規分布の累積分布関数です。rmsnorm は平均を引かず、2乗の平均の平方根だけで割ります (microgpt は γ も省いています)。GELU は u が大きい正の数ならほぼ u , 大きい負の数ならほぼ 0 で、0 の近くではなめらかに曲がります。ReLU はその曲がりを角にしたものです。GPT-2 の掛け算は $y = Wx + b$ (b がバイアス) ですが、microgpt は $y = Wx$ だけです。

付録B 用語のまとめ

用語	元の言葉	この冊子での意味
LLM	large language model	大規模言語モデル。ChatGPT や Claude など
トークン	token	AI が1回に予想する単位。microgpt では1文字
トークナイザ	tokenizer	文字と番号を行き来する規則
語彙	vocabulary	扱うトークンの種類。小さい例では7種類
BOS	beginning of sequence	名前の始まりと終わりを示す印
重み (パラメータ)	weight (parameter)	練習で調整される数。この冊子では「つまみ」
ハイパーパラメータ	hyperparameter	練習の前に人間が決める設定の数
黑板 (残差の流れ)	residual stream	位置ごとの4つの数。書き足していく
wte / wpe	token / position embedding	文字の表 / 位置の表 (最初のメモ)
rmsnorm	root mean square normalization	4つの数の大きさをそろえる
Attention	attention (注意)	前の位置の黑板を見に行くしくみ
q / k / v	query / key / value	問い合わせ / 見出し / 中身
softmax	softmax	得点を、合計1の割合 (確率) に直す
MLP	multi-layer perceptron	16個の検出器と書き足し (fc1, fc2)
ReLU	rectified linear unit	負なら0にする (点灯するかどうか)
lm_head	language model head	黑板を読んで7つの得点を出す表
ロジット	logit	softmax の前の得点
損失	loss	正解につけた確率の罰点 ($-\ln p$) の平均
交差エントロピー	cross-entropy	この損失の数学での名前
パープレキシティ	perplexity	平均して何択で迷っているか
勾配	gradient	つまみを少し増やしたときの、損失の増え方の割合
勾配降下法	gradient descent	勾配と反対の向きに少しずつ回す方法
自動微分	automatic differentiation	計算の記録をさかのぼって勾配を求めるしくみ

用語	元の言葉	この冊子での意味
誤差逆伝播法 連鎖律	backpropagation chain rule	連鎖律で出力側から勾配を伝える方法 影響の伝わり方（回転比）を掛け合わせる決まり
学習率 Adam 生成	learning rate adaptive moment estimation generation	1回に回す幅の目安（0.01 → 0） 勢いと幅そろえを使う回し方 学習後の AI に新しい名前を作らせること
温度	temperature	得点を割って、確率をとがらせたり、 ならしたりする設定
貪欲法	greedy decoding	いつもいちばん確率の高い文字を選ぶ 方法
KV キャッシュ	KV cache	前の位置の k と v をとっておいて使い 回すしくみ
幻覚（作り話）	hallucination	もっともらしいが事実でないものを作 ること
層	layer	Attention と MLP の1組. 大きな AI はこれを何回も重ねる
ヘッド GPU	head graphics processing unit	Attention を分けて行う組 同じ種類の計算を大量に同時に行える 装置
事前学習	pre-training	大量の文章で次のトークンを当てる練 習
RLHF	reinforcement learning from human feedback	人間の評価をもとにした追加の学習
スケーリング則	scaling law	つまみやデータを増やすと損失が規則 的に下がる傾向
解釈可能性	interpretability	AI の中で何が起きているかを読み解 く研究

付録C 数値の求め方とプログラムの入手先

- **学習の設定**：学習データは emma, ava, mia, anna の4つ。黒板の幅 4，一度に読める長さ 5，ヘッドの数 1，練習 300 回，乱数の種 42。冊子『microgpt を読む』5節の小さい例，冊子『小さなAIの頭の中をのぞく』と同じ設定です。学習1回目の損失 1.9535，300回目の損失 0.3563，学習後の wte の a の行 (+0.119, -0.207, +0.194, +0.183) が一致することで，同じ学習になっていることを確かめました。
- **2章の数値**：学習後の重みで，gpt() と同じ計算を普通の小数で書き直したプログラム (compute_part2.py) で，各位置の途中の計算をすべて求めました。
- **3章の数値**：同じ学習を行いながら，各回の後のつまみで18問の平均の罰点と各場面の確率を求めました (compute_part3.py)。1回目の勾配の数値での確かめは，つまみを 0.001 だけ動かして求めました。
- **4章の数値**：学習後，元のプログラムと同じ方法 (random.choices, 温度 0.5) で20個を作りました (compute_part4.py)。温度を変えた20個は，学習直後の乱数の状態から作り直しました。1000個の集計と学習途中の生成には，乱数の種 7 を使いました（学習の乱数には影響させていません）。

- **5章の数値**：つまみの数は公表されている構造から数え、元のデータの大きさは `names.txt` から数えました (`compute_part5.py`)。GPT-2 と GPT-3 の大きさは冊子『microgpt を読む』8.1節、GPT-3 の学習データの量は Brown ほか (2020) によります。
- **図**：`mkfigs_anatomy.py` で作りました。同じ内容のスライド (パワーポイント, 第1~5部) も作成しています。
- **microgpt のプログラ**
ム：<https://gist.github.com/karpathy/8627fe009c40f57531cb18360106ce95> (2026年9月30日取得版)。ライセンスの表示がないため、この冊子にはプログラムを載せていません。