そうぇり　もぷもぷ

～ミニマリズム人工言語のボットは作れるか～

多分言語創作と計算機が好きな人に向けた資料

soweli Mopumopu

- Twitter 上のトキポナボット
  - @soweli_mopumopu
  - 2019 年からうごいている

soweli Mopumopu

- 15 分おきに何か言葉を発します
- 会話もできます　たのしいね

なんの話をするの

- soweli Mopumopu はどうやって動いてるのか？
  - 実はすこし高度なことをしている
- もぷもぷの技術について説明します

トキポナとは

- みんなが知ってるミニマリズム人工言語
  - 語彙数: 120 語
  - 習得がかんたん　たのしい!
  - できる人が割と多い　たのしい!
- ごくわずかな知らない人のために
  - 初級トキポナ文法簡介
  - 2 分 40 秒で世界一簡単な言語を紹介して伝授する
  - トキポナレッスン1「トキポナってなに」
  - jan Pije's lessons (注)
  - 【日本語訳】toki pona li toki pona - トキポナソング

トキポナをモデル化する

- トキポナをコンピュータで扱いたい！
  - コンピュータには言語がわからぬ…
- “文” を数理モデル化しよう！
  - 確率的言語というのを考えます

確率的言語

- 「その文が起きる確率」というのを考える
  - 文の確率 $P(\text{文})$ を考える
  - $P(\text{toki pona li toki pona!})$ とか
- なにがうれしいの？

確率的言語

- 次の 2 つの文 A, B はどちらが “良い” 文だろうか？
  - 文 A: toki pona li toki pona.
    - 「トキポナは良い (pona) 言語 (toki)」
  - 文 B: a akesi ala alasa ale.
    - 非文
    - 明らかに A のほうがよい
    - $P(A) > P(B)$ となるはず
  - 確率的言語は文法を扱えるかも？

確率的言語

- 次の 2 つの文 A, B はどちらが “良い” 文だろうか?
  - 文 A: ma tomo Tokijo li lon ma Nijon.
    - 「東京は日本にある」
  - 文 B: ma tomo Lanten li lon ma Tosi.
    - 「ロンドンはドイツにある」
    - A は正しいが, B は間違い
    - $P(A) > P(B)$ となるはず
  - 確率的言語は意味や常識を扱えるかも?

確率的言語

- 確率で言語を近似することができれば，文法や意味を捉えたモデルを作成できるのでは？
  - “良い文”・“悪い文” を定性的に評価できる
    - チャットボットも作れる
  - もっと広く言語処理の様々な場面で使われている
    - 機械翻訳などもこの考えを元に作られている
- 文の確率をどうやって計算するの？

言語モデル

- 文に確率を与えるモデルのこと
- つまり,
  - 文 $w_1^n = w_1 w_2 \cdots w_n$ を入力して,
  - 文の確率 $P(w_1^n)$ を計算・出力する関数っぽいの
- どのように計算する?
  - そもそも入力の長さが文ごとに違うしつらい…

分解する

- 確率論の乗法定理を用いて, $P(w_1^n)$ を分解

$$
\begin{aligned}
P(w_1^n) &= P(w_1, w_2, \cdots, w_n) \\
&= P(w_1) P(w_2, w_3, \cdots, w_n | w_1) \\
&= P(w_1) P(w_2 | w_1) P(w_3, w_4, \cdots, w_n | w_1, w_2) \\
&= P(w_1) P(w_2 | w_1) P(w_3 | w_1^2) P(w_4^n | w_1^3) \\
&\cdots \\
&= P(w_1) \prod_{i=2}^{n} P(w_i | w_1^{i-1})
\end{aligned}
$$

- 1 単語ずつ計算してかければ文の確率になる!

$P(w_i|w_1^{i-1})$ を計算する

- 1 番目から $i-1$ 番目の単語がわかってて, i 番目の単語が起こる確率
- $P(\text{li}|\text{toki pona}) > P(\text{wile}|\text{toki pona})$
- Mopumopu では, ニューラルネットを用いている
  - ニューラル言語モデルっていう

Transformer モデル

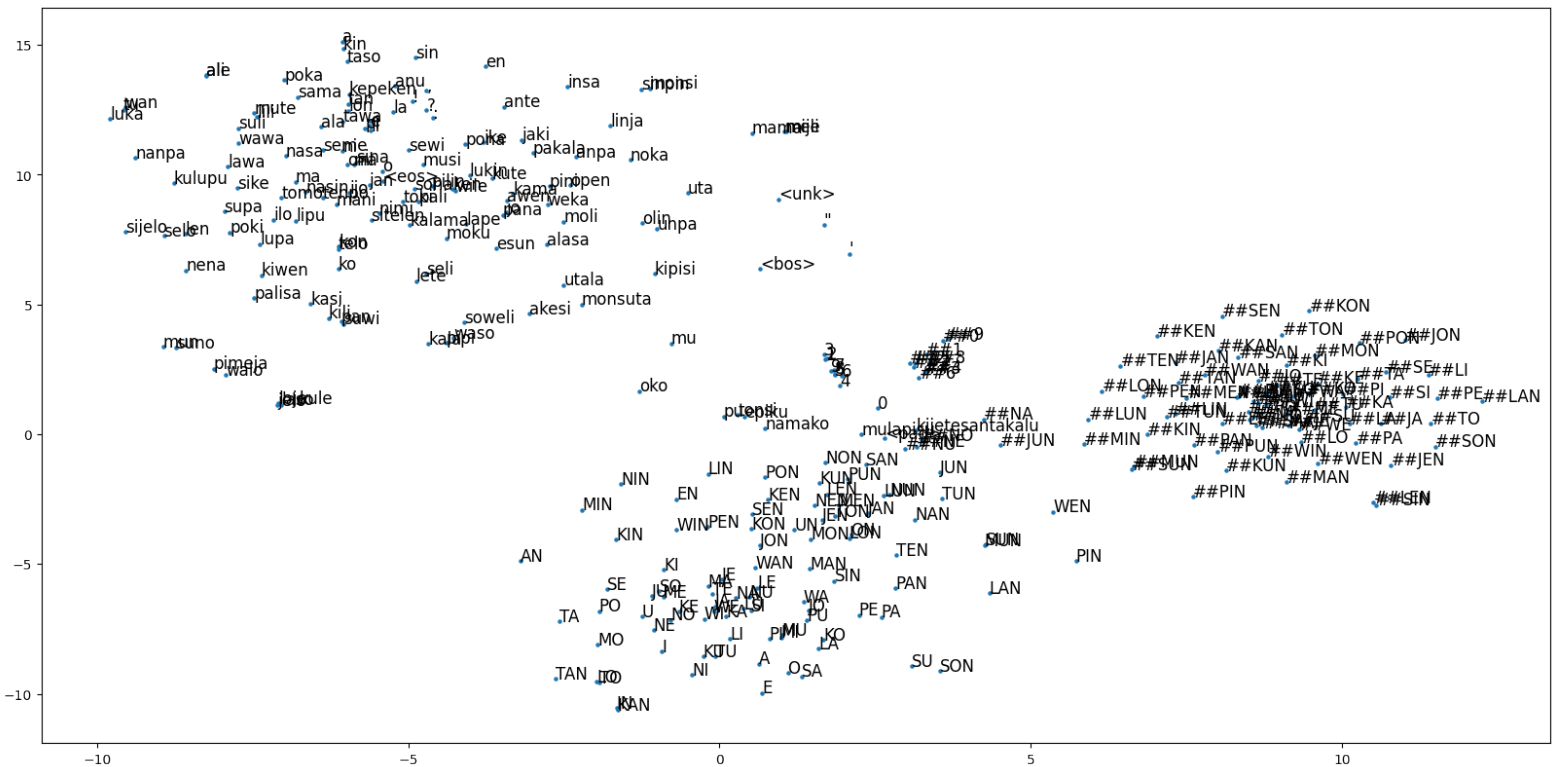
- Mopumopu で採用したニューラルモデル
  - 2017 年に Google の人たちが考案した
    - Google 翻訳のなかみもこれ（だと思う）
- 次からのスライドで細かいことを説明します
  - 注: ここで説明するのは, 一般によく言われている Transformer Encoder-Decoder モデルのことではなく, Transformer 言語モデルのことです.

Transformer モデル

- Transformer モデルは次の部分からなっている
  - 単語 Embedding 層
  - 位置 Embedding 層
  - Self-attention 層
  - Feed-forward 層

単語 Embedding 層

- 単語をベクトルに変換する
  - Mopumopu では 1024 次元
  - 逆（ベクトル-> 単語）もできる

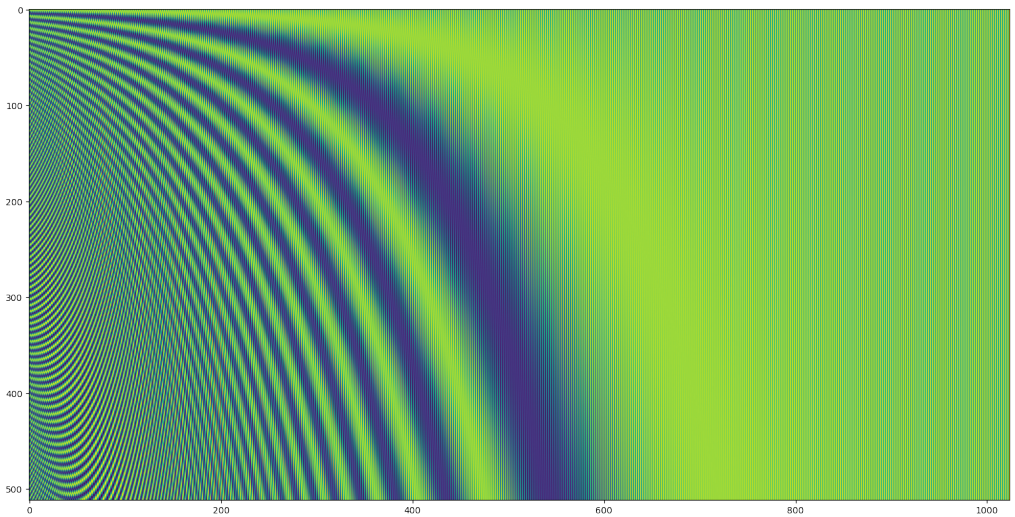


位置 Embedding 層 (難しい)

- 位置（何番目の単語か）をベクトルで表す
  - 三角関数の回転を利用している
  - t 番目の i 次元が

$$\boldsymbol{p}_t^{(i)} = \begin{cases} \sin\left(\omega_k t\right) & (i = 2k) \\ \cos\left(\omega_k t\right) & (i = 2k+1) \end{cases}$$

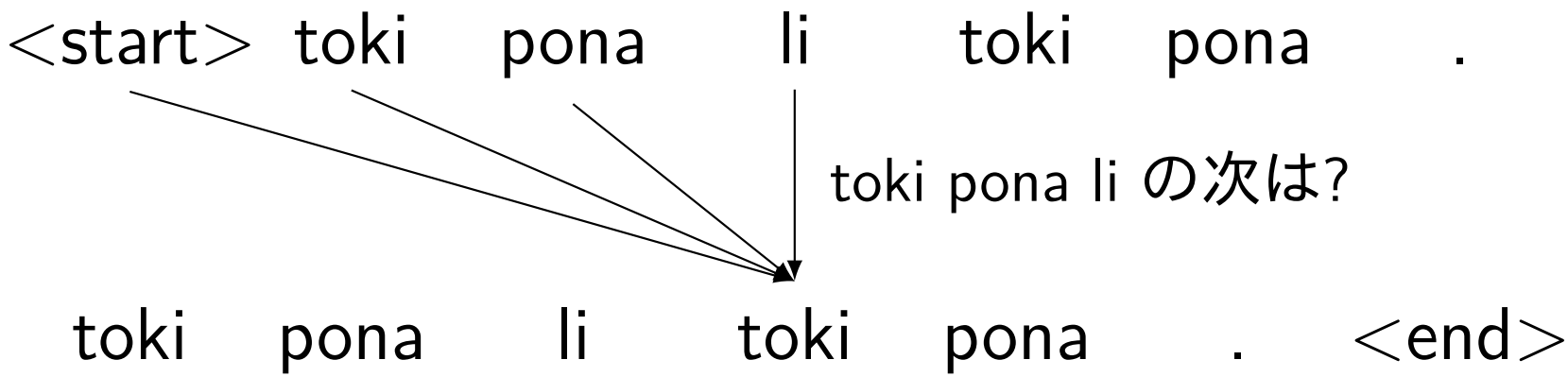
  - 細かいことはわかんなくてもいいよ



← こんなん

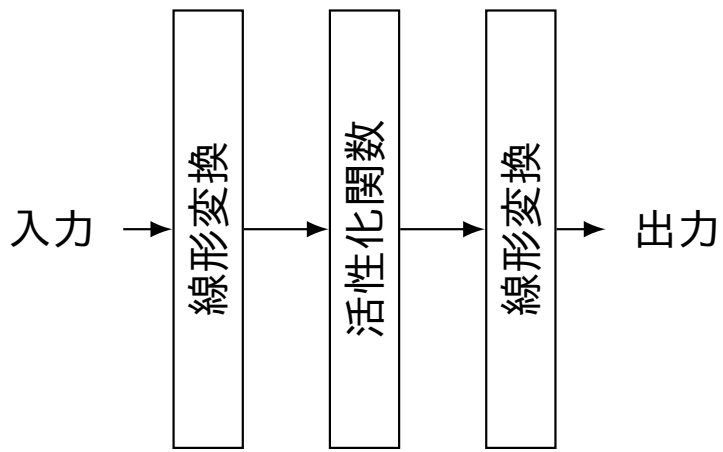
Self-attention 層 (難しい)

- 前の単語の情報を足し合わせ, 次の単語を予測
  - 文脈から意味や構造を捉えて, 次に来そうな単語を決めていく
  - Multi-head Attention とか, Residual Connection とか, Layer Normalization とか, 細かいことは調べて



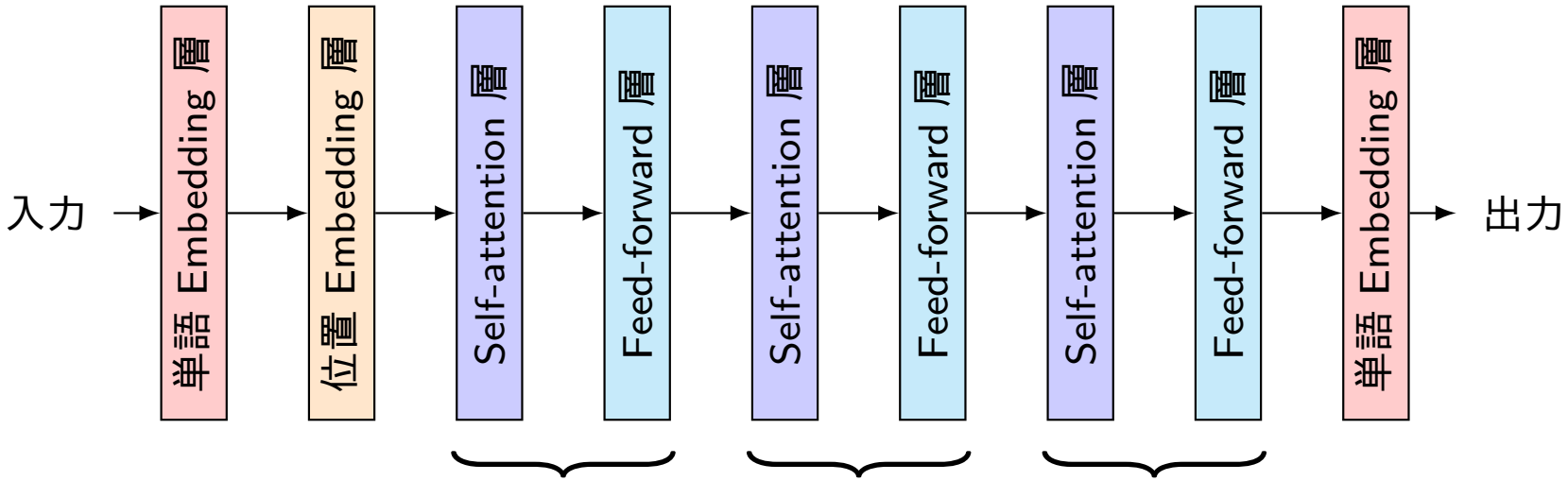
Feed-forward 層

- 足し合わせるだけだと厳しいので, 非線形変換をする
  - 活性化関数が非線形変換になっているので, 複雑な変換ができる(多分)



Transformer モデル

- それをこんな感じで組み合わせると



- なんと言語モデルができる
  - このへんの話はとてもむずかしいので, わからなくてもいいです（そもそもわかるように書いてない）
    - 自分はやってることわかるまで半年かかった

どのように学習するのか？

- モデルの確率分布を言語の確率分布に近づける
  - モデルと言語の相対エントロピーを小さくする

言語のエントロピー

- 言語 L の文 w_1^n の起こる確率は $P(w_1^n)$
- 文 w_1^n のエントロピーは

$$-P(w_1^n)\log P(w_1^n)$$

- 言語全体でのエントロピーは

$$H(L) = -\sum_{w_1^n} P(w_1^n)\log P(w_1^n)$$

  - これを言語のエントロピーと呼びます
    - 言語を文の情報源とみなしたときの，情報源全体の不確実さを表している

相対エントロピー

- 確率分布 P と Q の距離を表す尺度

$$D(P||Q) = \sum_i P(x_i) \log \frac{P(x_i)}{Q(x_i)}$$

  - 大きいほど P と Q は離れている
  - カルバック･ライブラー距離という
    - これ, 距離の公理を満たしてないんだよな

クロスエントロピー

- 言語 L の確率分布 P_L と言語モデル M の確率分布 P_M を近づけたい!
  - P_L と P_M の相対エントロピーを小さくしよう!

クロスエントロピー

$$
\begin{aligned}
D(L||M) &= \sum_{w_1^n} P_L(x_i) \log \frac{P_L(x_i)}{P_M(x_i)} \\
&= -\sum_{w_1^n} P_L(x_i) \log P_M(x_i) + \sum_{w_1^n} P_L(x_i) \log P_L(x_i) \\
&= \underbrace{H(L, M)}_{\text{クロスエントロピー}} - \underbrace{H(L)}_{\text{言語のエントロピー}}
\end{aligned}
$$

- クロスエントロピーを小さくすればよい!
  - どうやって小さくする?

最急降下法

- クロスエントロピーを微分した方向の逆にモデルのパラメータを動かしていけば, 小さくなっていくのでは?
  - 「標高が低い方に進めば下山できるのでは?」

    遭難するのでやってはいけない

  - 言語モデルが学習できる!

最急降下法

- クロスエントロピー

$$H(L, M)$$

- その微分（勾配という）

$$G = \nabla H(L, M)$$

- モデルのパラメータ

$$\Theta$$

- つまり，適当な学習率 α で，こうすればよい

$$\Theta := \Theta - \alpha G$$

- ほんとにこんなんで学習できるのか？
  - ▷ 厳しいので工夫する

Adam アルゴリズム

- 学習中の時点 t の勾配 G_t とし,

$$m_t = \beta_1 m_{t-1} + (1-\beta_1) G_t \quad \text{（向きの調整）}$$
$$v_t = \beta_2 v_{t-1} + (1-\beta_2) G_t^2 \quad \text{（大きさの調整）}$$

  を求めて,

$$\Theta := \Theta - \frac{\alpha m_t}{\sqrt{v_t} + \epsilon}$$

  を計算する. (ϵ は 0 除算を回避するためのマシンイプシロンだと思う)

- これだと割とうまく行くことが知られている
  - Mopumopu でもこれを使っている

自己回帰生成

- 言語モデルが学習できたら, 文の生成をします

ランダムサンプリング

ランダムサンプリングは厳しい

top-p サンプリング

PyTorch

nymwa/ponapt

データセット

トークナイズとか，前処理とか

- “toki!” みたいな文はそのまま処理できない
  - “ toki ！ ” みたいに単語や記号ごとに分割したい

実際に動かす

言語モデルでチャットボットをつくる

- 会話のデータがないのでちょっと無理がある
- すこし怪しいことをしている
  - 入力文 *S* を受け取る.
  - 入力文に引用符を付けて, “*S*” みたいにする
  - “*S*” “ に続く文を生成させる
- あんまり会話にはなってないけど, トキポナなのでなんとなく会話として解釈される感じがある

Twitter API を使う

おわりに

$$\equiv > \omega < \equiv$$

pona tawa sina!