テキスト埋め込みからのテキスト復元における予測制御の援用の効果検証

三好優輝 [1], 宮岡祐弥 [1], 井上正樹 [1]
[1] 慶應義塾大学

概要

テキスト埋め込みから元のテキストを復元する方法を, Vec2Text[1] とは別のアプローチで考えた. そのアプローチは, テキスト生成において予測制御の考えを用いる方法である. 予測制御によるテキスト生成を用いると, Greedy Search によりテキスト生成した場合よりも性能が向上した. 具体的には, 2 つのテキストの内容が一致する割合が最大 0.120 だけ向上した.

1 序論

テキスト埋め込みモデルとは, テキストの内容を単一のベクトルに抽出するエンコーダモデルである. テキスト埋め込みでは, 2 つのテキストの内容の類似度計算が可能であり, 情報検索・質問応答・バイテキストマイニングなど様々な NLP タスクに利用されている [2].

この抽出されたテキスト埋め込みから元のテキストを復元することは一般に難しい. 文献 [1] では, テキスト埋め込みからテキストを復元する手法として Vec2Text という手法が提案された. Vec2Text では, 2 つのモデルが用いられ, 1 つ目のモデルでテキスト埋め込みから元のテキストの仮説を生成し, 2 つ目のモデルで生成したテキストの修正を行なう. そして, 2 つ目のモデルにより, テキストの修正を反復的に行なうことで, 高い精度で元のテキストを復元することが可能になっている.

本研究では, Vec2Text とは異なるアプローチにより, テキスト埋め込みからテキストを復元することを目指した. 具体的には, Vec2Text の 2 つ目のモデルを使用せず, 1 つ目のモデルのテキスト生成方法に予測制御の考え方を用いる方法を考えた.

本稿の構成を示す. 2 章でテキスト埋め込みから元のテキストを復元する “テキスト復元モデル” の構築について, 3 章で予測制御の考え方を用いたテキスト生成方法とその効果について. 4 章で結論を述べる.

表記: 語彙の集合を $\mathbb{V}$ とおき, トークンは $w \in \mathbb{V}$ で表す. そして, 単一および複数のトークンの列をテキストと呼び, $x = \{w_i\}_{i=1}^{n} \in \mathbb{V}^n$ と表す. テキスト埋め込みモデルは関数 $\phi : \mathbb{V}^n \to \mathbb{R}^d$ で表す.

2 準備: テキスト復元モデルの構築

本章では, テキスト埋め込みから元のテキストを復元するためのモデルとして構築したテキスト復元モデルについて述べる. テキスト復元モデルは, テキスト埋め込み $\boldsymbol{e}_{\text{ref}} \in \mathbb{R}^d$ と生成中のトークン列 $\{w_i\}_{i=1}^{n} \in \mathbb{V}^n$ を入力とし, 後続トークンの出現確率分布 $P \in \mathbb{R}^{|\mathbb{V}|}$ を出力するモデルである. 2.1 節では, テキスト復元モデルの訓練について, 2.2 節では, 構築したテキスト復元モデルの性能について述べる.

2.1 テキスト復元モデルの訓練

モデル構造 構築したテキスト復元モデルのモデル構造を図 1 に示す. このモデルは, エンコーダ・デコーダモデルからエンコーダ部分を取り除いたものである. そして, エンコーダから交差注意機構への入力の代わりとして, テキスト埋め込み $\boldsymbol{e}_{\text{ref}} \in \mathbb{R}^d$ を入力した. ただし, テキスト埋め込みの次元 d は, デコーダ部分の潜在ベクトル $\boldsymbol{h}_i$ の次元と一致させる必要がある.

本研究では, 基盤のエンコーダ・デコーダモデルとして “sonoisa/t5-base- japanese” [1)] を使用し, その潜在ベクトルと同次元のテキスト埋め込みを得られる “sonoisa/sentence-bert-base-ja-mean-tokens-v2” [2)] というテキスト埋め込みモデルを使用した.

1) https://huggingface.co/sonoisa/t5-base-japanese
2) https://huggingface.co/sonoisa/sentence-bert-base-ja-mean-tokens-v2

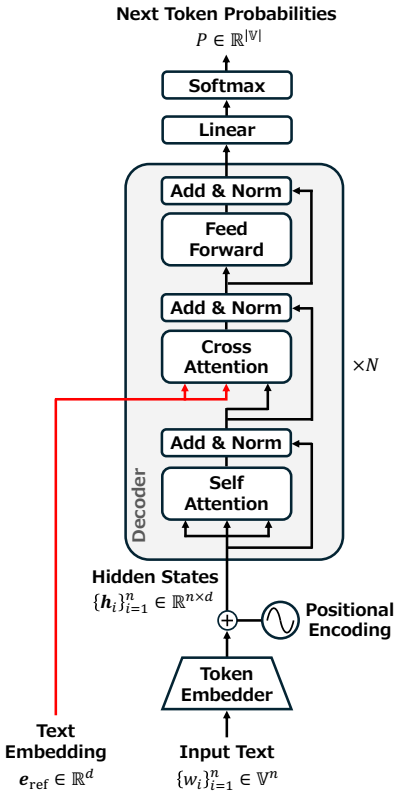

図 1　構築したテキスト復元モデル

表 1　訓練時のハイパーパラメータの設定

ハイパーパラメータ	値
LoRA 低ランク行列 A の学習率 ρ_A	1.75×10^{-6}
LoRA 低ランク行列 B の学習率 ρ_B	$\rho_A \times 2^4$
LoRA 低ランク行列 A のランク	64
バッチサイズ	8
エポック数	2

訓練データ　訓練データとして, 文献 [3] で使用されている JSNLI データセット[3] の訓練セットを使用した. これは合意関係認識タスクの訓練に使用されるデータセットである. 本研究では, データセット内の前提文と仮説文の合計 106 万 6010 個のテキストを用いて, テキスト埋め込み $\boldsymbol{e}_{\text{ref}}$ とトークン列 $\{w_i\}_{i=1}^n$ の 2 入力と, 後続トークン w_{n+1} の出力のデータセットを構築した.

訓練方法　テキスト復元モデルの訓練には, LoRA[4] を用いた. デコーダ内の自己注意機構・交差注意機構の全ての線形層に LoRA 層を挿入し, LoRA 層のパタメータのみを訓練した. 訓練時のハイパーパラメータの設定を表 1 に示す. 本研究では, LoRA の低ランク行列 A, B を異なる学習率で訓練する LoRA+[5] という手法を使用した.

3)　https://huggingface.co/datasets/llm-book/jsnli

2.2　構築したモデルの性能検証

本節では, 2.1 節で構築したテキスト復元モデルの性能について述べる.

検証方法　構築したテキスト復元モデルの性能として, Greedy Search により 1 からテキスト復元を行なった場合に, どれだけのテキストをどれほどの精度で復元できるのかを調べた.

本研究では, 生成されたテキスト x_{gen} が元のテキストとどの程度異なるかの指標として, 生成テキストのテキスト埋め込み $\phi(x_{\text{gen}})$ と目標のテキスト埋め込み $\boldsymbol{e}_{\text{ref}}$ のユーグリッド距離

$$d_2(\phi(x_{\text{gen}}), \boldsymbol{e}_{\text{ref}}) \coloneqq \|\phi(x_{\text{gen}}) - \boldsymbol{e}_{\text{ref}}\|_2 \tag{1}$$

を用いた[4] .

検証データには, 日本語言語理解ベンチマーク JGLUE[6] の JSTS データセットの検証セットを用いた. JSTS は 2 つのテキストの意味的類似度計算のデータセットであり, 合計 2914 個のテキストを検証に使用した. そして, それぞれのテキストのテキスト埋め込み $\boldsymbol{e}_{\text{ref}}$ から元のテキストをどの程度復元できるかを検証した.

検証結果　まず, 構築したテキスト復元モデルの性能について述べる前に, (1) 式で定義した指標 $d_2(\phi(x_{\text{gen}}), \boldsymbol{e}_{\text{ref}})$ の値とテキストの復元度の関係について調べた. テキスト距離 $d_2(\phi(x_{\text{gen}}), \boldsymbol{e}_{\text{ref}})$ の振る舞いの例を表 2 に示す. 表 2 より, 2 つのテキストの内容が類似しているときほど, テキスト距離 $d_2(\phi(x_{\text{gen}}), \boldsymbol{e}_{\text{ref}})$ が小さくなることが確かめられた. そして, テキスト距離 $d_2(\phi(x_{\text{gen}}), \boldsymbol{e}_{\text{ref}})$ が 5 以下であれば, おおよそ 2 つのテキストの内容は一致することが分かった.

構築したテキスト復元モデルにより生成されたテキスト x_{gen} と元のテキストの距離 $d_2(\phi(x_{\text{gen}}), \boldsymbol{e}_{\text{ref}})$ の分布を図 2 に示す. テキスト距離の平均 $\mathbb{E}(d_2)$ は 13.0 であり, テキスト距離が 5 以下となる割合 $p_{d_2 \leq 5}$ は 0.041 となった.

3　テキスト復元モデルの出力制御

本章では, 2 章で構築したテキスト復元モデルの性能を向上させるためのテキスト復元モデルの出力制御について述べる. まず, 3.1 節で, 予測制御の考え方を用いたテキスト生成のアルゴリズムについて説

4)　NLP においては, ベクトルの違いの指標としてコサイン類似度を用いることが多いが, 本研究ではユーグリッド距離を使用した.

表 2　テキスト距離 $d_2(\phi(x_{\text{gen}}), \boldsymbol{e}_{\text{ref}})$ の値の例

元のテキスト	生成テキスト x_{gen}	$d_2(\phi(x_{\text{gen}}), \boldsymbol{e}_{\text{ref}})$
子供たちが 遊んでいます。	子供たちが 遊んでいます。	0.0
スケート ボーダーが 手すりを 滑っています。	スケート ボーダーは 手すりを 滑っています。	2.42
3 頭の牛が 草の上に。 並んで 立っています。	草の列に 3 頭の牛が 立って います。	5.29
棚の中に 犬がいます。	棚に犬が 棚の棚を 棚棚の中に います。	7.03
壁の前に ソファーと テーブルが 置いてあります。	テーブルの前に ソファーが あります。	10.0

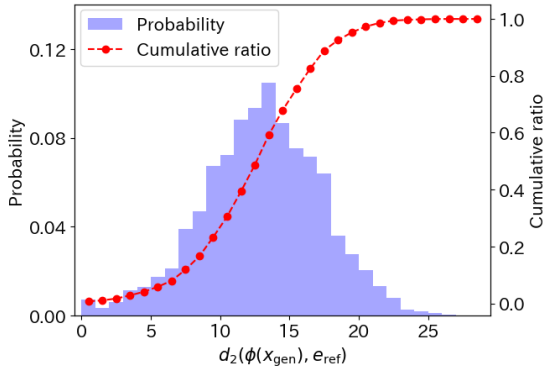

図 2　構築したテキスト復元モデルの性能

明する. そして, 3.2 節では, 図 2 の結果をベースラインとし, テキスト生成方法によってどれだけ性能が向上するかを検証する.

3.1　予測制御を用いたテキスト生成

予測制御とは, 将来の出力を予測しながら, 現在の行動を決定する制御手法である. 本研究では, 予測制御の考え方を用いたテキスト生成によりテキスト復元の精度を上げることを考えた.

パラメータ　予測制御には主に 2 つのパラメータがある. 一つ目は予測ホライゾン H_{P} であり, 現在から何ステップ先までの出力を予測するかに対応している. 二つ目は制御ホライゾン H_{C} であり, 得られた予測から何ステップ先までの行動を決定するかに対応している.

本研究では, テキスト生成に予測制御を用いる上で, 予測ホライゾンと制御ホライゾンに加えて, 新たに予測ブランチ列 $\boldsymbol{b} \in \mathbb{N}^{H_{\text{P}}}$ とエージェント数 N_{a} と

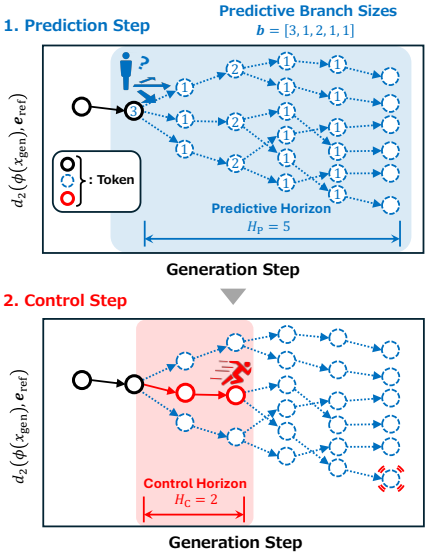

図 3　予測制御的なテキスト生成の概要 (1)

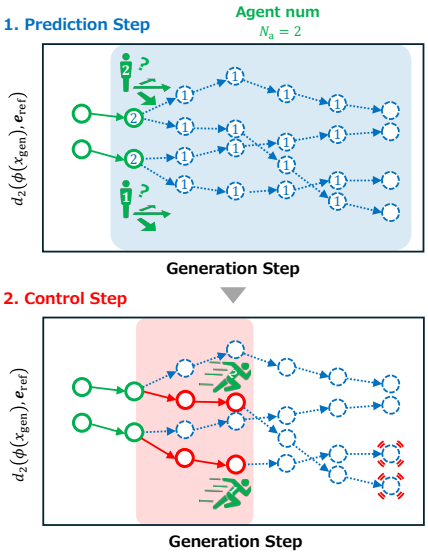

図 4　予測制御的なテキスト生成の概要 (2)

いうパラメータを導入した. まず, 予測ブランチ列 $\boldsymbol{b} = [b_1, b_2, ..., b_{H_{\text{P}}}]$ の概要を図 3 に示す. 予測ブランチ列 $\boldsymbol{b}$ の i 列目の要素 b_i は, 現時点から i ステップ先で予測する分岐路の数に対応している. 次に, エージェント数 N_{a} の概要を図 4 に示す. エージェント数 N_{a} は, 行動決定の際に候補として残しておく路の数に対応している.

アルゴリズム　本研究のテキスト生成アルゴリズムは, 図 3, 4 のように予測ステップと制御ステップの 2 つで構成される. このアルゴリズムの概要について説明する.

予測ステップでは, 入力トークン列から予測ブランチ列 $\boldsymbol{b}$ に従い, トークン列を分岐させて推論す

表 3　各メソッドのパラメータ設定

メソッド名	H_P	H_C	$\boldsymbol{b}$	N_a	$N_\mathrm{a} \times \prod_{b_i \in \boldsymbol{b}} b_i$
ベースライン	1	1	[1]	1	1
A	1	1	[3]	1	3
B	50	1	[3, 1, 1, ..., 1]	1	3
C	50	1	[3, 1, 1, ..., 1]	2	6
D	50	1	[4, 1, 1, ..., 1]	2	8
E	50	1	[3, 1, 1, ..., 1]	3	9
F	50	1	[4, 1, 1, ..., 1]	3	12
G	50	1	[8, 1, 1, ..., 1]	1	8
H	50	1	[4, 1, 1, ..., 1]	2	8
I	50	1	[3, 2, 1, ..., 1]	1	6
J	50	1	[3, 2, 1, ..., 1]	2	12

る. ただし, i ステップ先のトークンの推論では, 出現確率が上位 b_i までのトークンを選択する. そして, 最終的に, トークン長 H_P の後続トークン列を計 $N_\mathrm{a} \times \prod_{b_i \in \boldsymbol{b}} b_i$ 個だけ推論する.

制御ステップでは, 予測ステップで得られた計 $N_\mathrm{a} \times \prod_{b_i \in \boldsymbol{b}} b_i$ 個のトークン列から, 候補として残すトークン列を N_a 個だけ選択する. ここで, 候補として残すトークン列には, テキスト距離 $d_2(\phi(x_\mathrm{gen}), \boldsymbol{e}_\mathrm{ref})$ が小さいものを順に選択する. そして, 選択したそれぞれのトークン列から, H_C ステップ先までのトークン列を生成する.

この予測ステップと制御ステップを繰り返すことにより, N_a 個のテキストが生成できる. そして, 最終的に, N_a 個のテキストの中でテキスト距離 $d_2(\phi(x_\mathrm{gen}), \boldsymbol{e}_\mathrm{ref})$ が最小になるテキストを生成する.

3.2　予測制御によるテキスト生成の効果

3.1 節で, 予測制御を用いたテキスト生成について述べた. 本節では, その効果について述べる.

実験設定　メソッド A ～ J の 10 個のメソッドを考えた. 各メソッドのパラメータ設定を表 3 に示す. ここで, $N_\mathrm{a} \times \prod_{b_i \in \boldsymbol{b}} b_i$ は予測ステップにおいて推論するテキストの総数であり, テキスト生成にかかる計算量に対応している. また, ベースラインのパラメータ設定は, 2.2 節の検証の際の設定と同じである.

この実験では, 出力されるトークンの数は最大で 50 個となるように設定した. また, 予測ステップにおいて, 出現確率が 0.9 以上のトークンが存在する場合は, その生成ステップでは分岐をせず推論した.

実験結果　各メソッドにおけるテキスト距離の平均 $\mathbb{E}(d_2)$, テキスト距離が 5 以下となる割合 $p_{d_2 \leq 5}$ を表 4 に示す. 表 4 より, 最良の性能を示したメソッドは F であった. メソッド F ではテキスト距離が 5 以下となる割合 $p_{d_2 \leq 5}$ は 0.161 となり, ベースラインよりも 0.120 だけ高くなった. メソッド F のテキスト距離 $d_2(\phi(x_\mathrm{gen}), \boldsymbol{e}_\mathrm{ref})$ の分布を図 5 に示す. ベースラインの分布である図 2 と比較すると, メソッド F では分布が大きく左に寄り, テキスト距離が 1 以下となる割合 $p_{d_2 \leq 1}$ は約 0.02 だけ高いことが分かった.

表 4　各メソッドの効果の結果

メソッド名	$\mathbb{E}(d_2)$	$p_{d_2 \leq 5}$
ベースライン	13.0	0.041
A	10.7	0.056
B	9.3	0.107
C	8.8	0.126
D	8.4	0.147
E	8.5	0.144
F	8.1	0.161
G	8.1	0.155
H	8.4	0.147
I	10.0	0.080
J	9.1	0.103

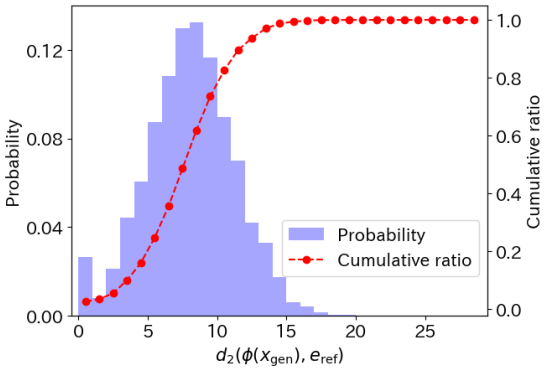

図 5　メソッド F の性能

そして, メソッド A ～ F より, $N_\mathrm{a} \times \prod_{b_i \in \boldsymbol{b}} b_i$ が大きいほど, おおよそ性能が向上した. ただし, メソッド D と E を比較すると, $N_\mathrm{a} \times \prod_{b_i \in \boldsymbol{b}} b_i$ が小さい D の方が E よりも性能が高くなった. また, $N_\mathrm{a} \times \prod_{b_i \in \boldsymbol{b}} b_i$ の値が同じメソッド G と H を比較すると, 1 ステップ目の予測ブランチ b_1 が大きい G の方が, エージェント数 N_a が大きい H よりも性能が高くなった.

また, メソッド B, C と I, J を比較すると, 2 ステップ目の予測ブランチ b_2 が大きい I, J の方が B, C よりも性能が低くなった.

4　結論

予測制御によるテキスト生成を用いると, Greedy Search よりもテキスト距離が 5 以下となる割合が最大 0.120 だけ向上した. また, 予測ステップで推論するテキスト数が同じ場合, 1 ステップ目の予測ブランチ b_1 が大きい方が, エージェント数 N_a が大きいものよりも性能が高くなることが分かった.

参考文献

[1] John Xavier Morris, Volodymyr Kuleshov, Vitaly Shmatikov, and Alexander M Rush. Text embeddings reveal (almost) as much as text. In **The 2023 Conference on Empirical Methods in Natural Language Processing**, 2023.

[2] Liang Wang, Nan Yang, Xiaolong Huang, Linjun Yang, Rangan Majumder, and Furu Wei. Improving text embeddings with large language models. **arXiv preprint arXiv:2401.00368**, 2023.

[3] 山田育矢ほか. 大規模言語モデル入門. 株式会社技術評論社, 2023.

[4] Edward J Hu, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, et al. Lora: Low-rank adaptation of large language models. In **International Conference on Learning Representations**.

[5] Soufiane Hayou, Nikhil Ghosh, and Bin Yu. Lora+: Efficient low rank adaptation of large models. In **Forty-first International Conference on Machine Learning**.

[6] 栗原健太郎, 河原大輔, 柴田知秀. Jglue: 日本語言語理解ベンチマーク. 自然言語処理, Vol. 30, No. 1, pp. 63–87, 2023.