

近傍の事例を用いた 非自己回帰生成

丹羽彩奈 高瀬翔 岡崎直觀

東京工業大学

高品質な文を速く生成する モデルの開発に向けて

- 生成速度を向上させるのに非自己回帰モデル(NAR)が有望

	非自己回帰モデル(NAR)	自己回帰モデル(AR)
生成速度	並列生成	逐次生成
生成される文の品質	ARには劣る傾向	高い

Diagram illustrating the architecture of NAR and AR models:

- NAR Model:** A sequence of tokens [A, B, D, ...] is fed into a "NARデコーダ". The output of the NAR decoder is "[PAD]" (padding). The NAR model is labeled as "並列生成" (parallel generation).
- AR Model:** A sequence of tokens [BOS, A, B, C, ...] is fed into an "ARデコーダ". The output of the AR decoder is a sequence of tokens [BOS, ..., C]. The AR model is labeled as "逐次生成" (sequential generation).

- NARの性能を上げるため、反復的デコーディングなどが提案されているが、**生成速度の優位性を維持しつつ自己回帰モデルと同等の生成品質に引き上げることは達成できていない** 2

本研究の概要

目的

非自己回帰モデルによる生成文の品質をより少ない生成回数で向上

アイデア

近傍の事例を非自己回帰モデルに組み込む

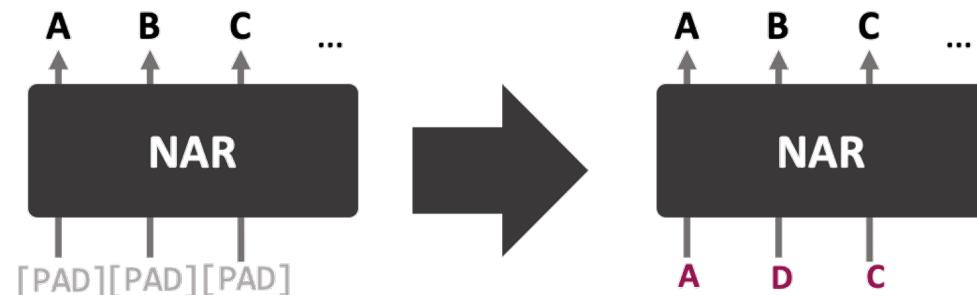
近傍の事例×非自己回帰モデルに取り組んだ既存研究はない

① デコーダの入力を近傍の事例で初期化

- Levenshtein Transformerをベースとし、近傍の事例を繰り返し編集して文を生成

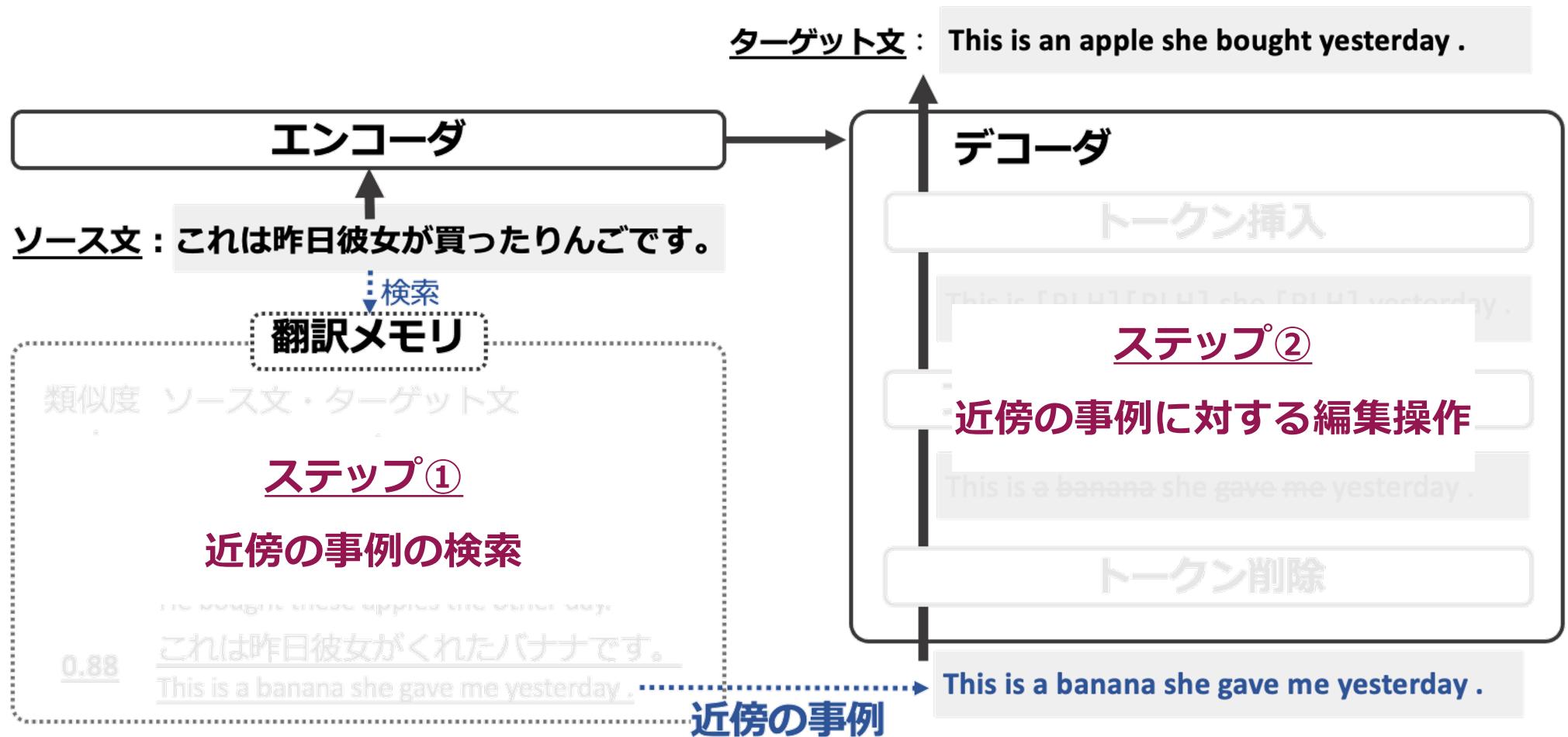
② 近傍の事例に対する編集操作を効率的に学習するための学習方策を設計

全てのトークンを無から生成するよりも容易に文を生成できる👍



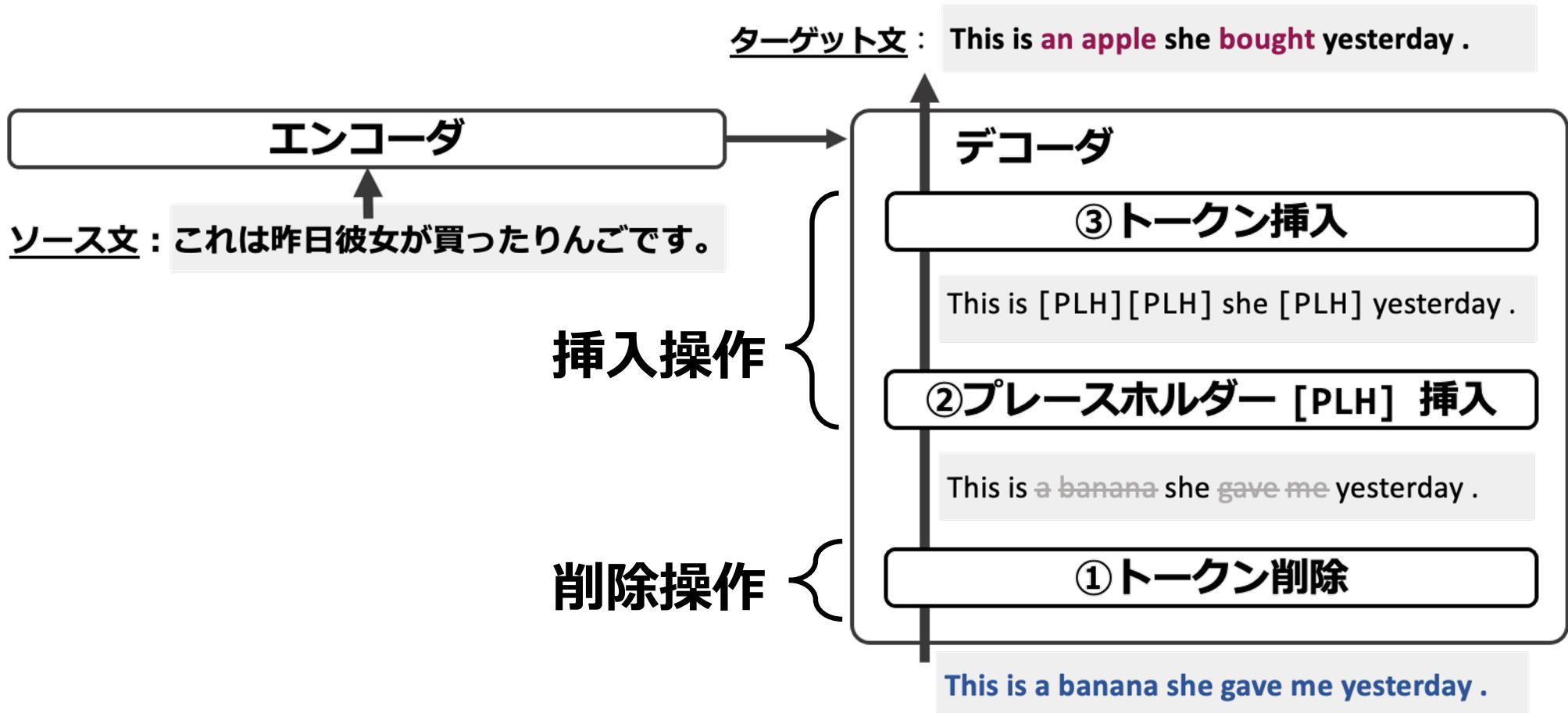
提案手法: *NeighborEdit*

ソース文と近傍の事例を用いてターゲット文を生成する系列変換タスク
(今回は翻訳タスク)



近傍の事例に対する編集操作

編集操作に基づく非自己回帰モデルLevenshtein Transformerに着想を得たモデル

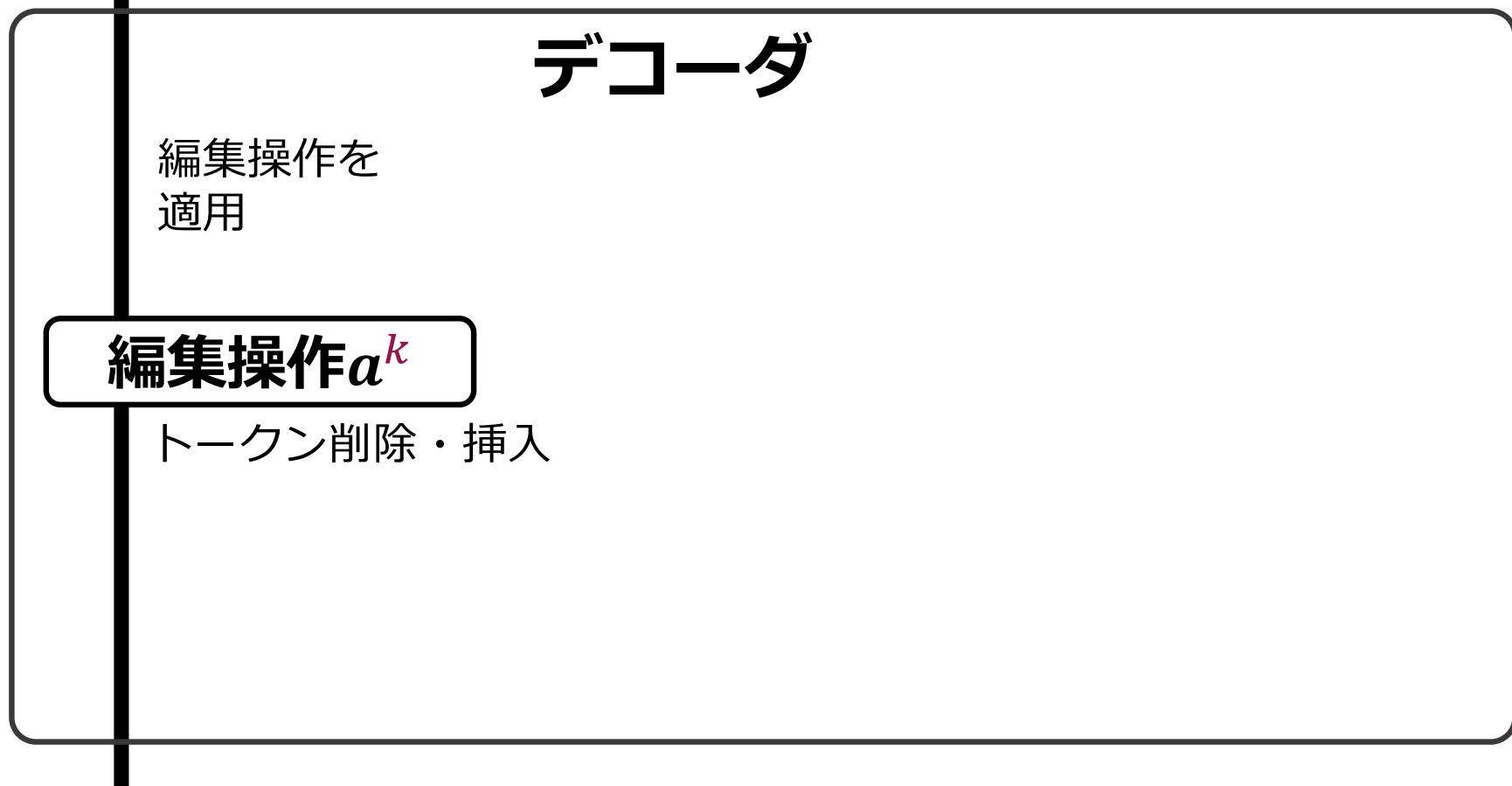


デコーダの最初の入力に**近傍の事例**を用いる

編集操作によるデコーディング

k 回目のデコーディング

$y^k = \varepsilon(y^{k-1}, a^k)$: 系列 y^{k-1} に編集操作 a^k を行なった系列

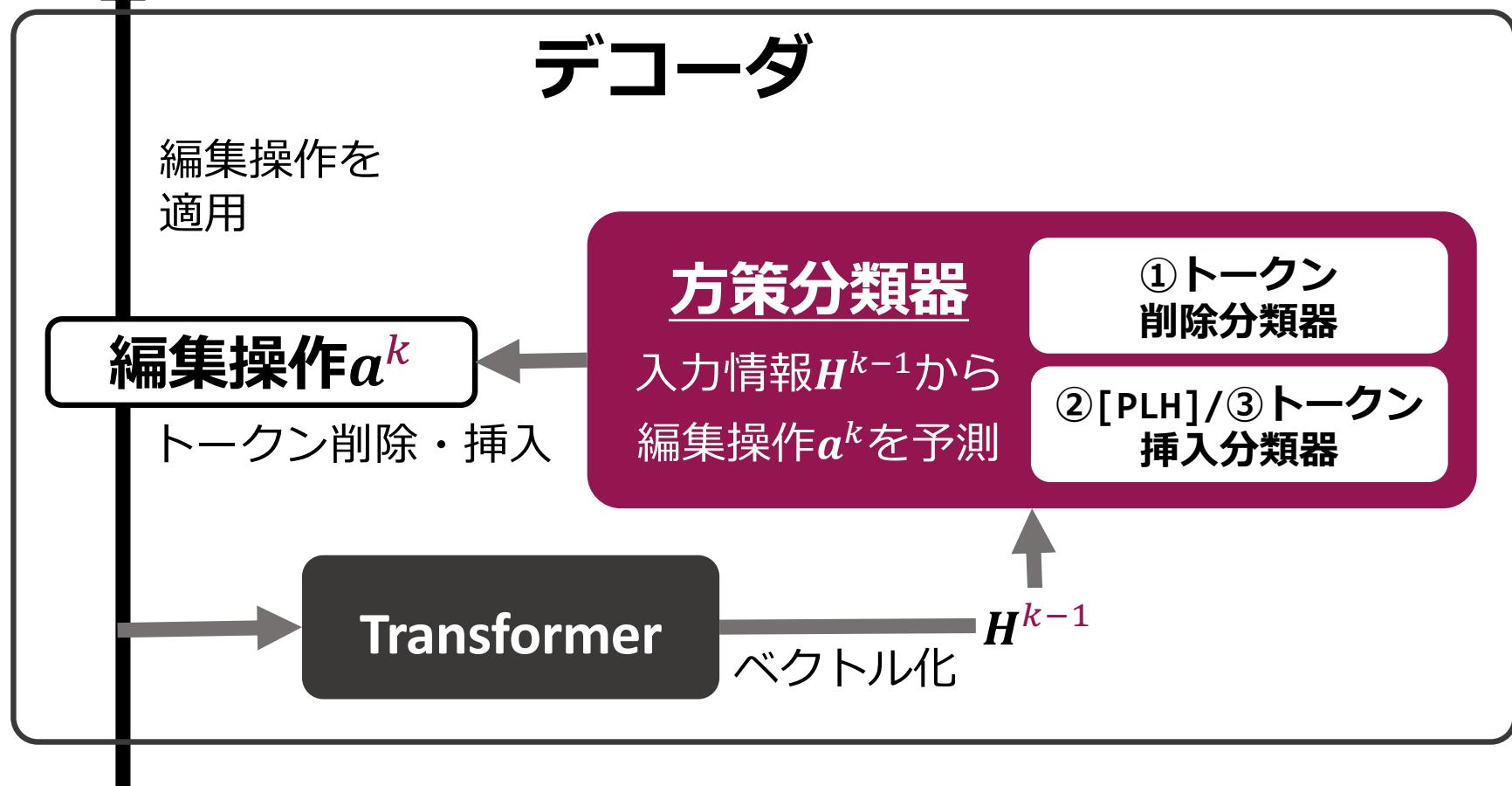


y^{k-1} : 前ステップまでに編集された系列
 $k = 0$ では近傍の事例

編集操作によるデコーディング

k 回目のデコーディング

$$y^k = \varepsilon(y^{k-1}, a^k) \quad : \text{系列 } y^{k-1} \text{ に編集操作 } a^k \text{ を行なった系列}$$



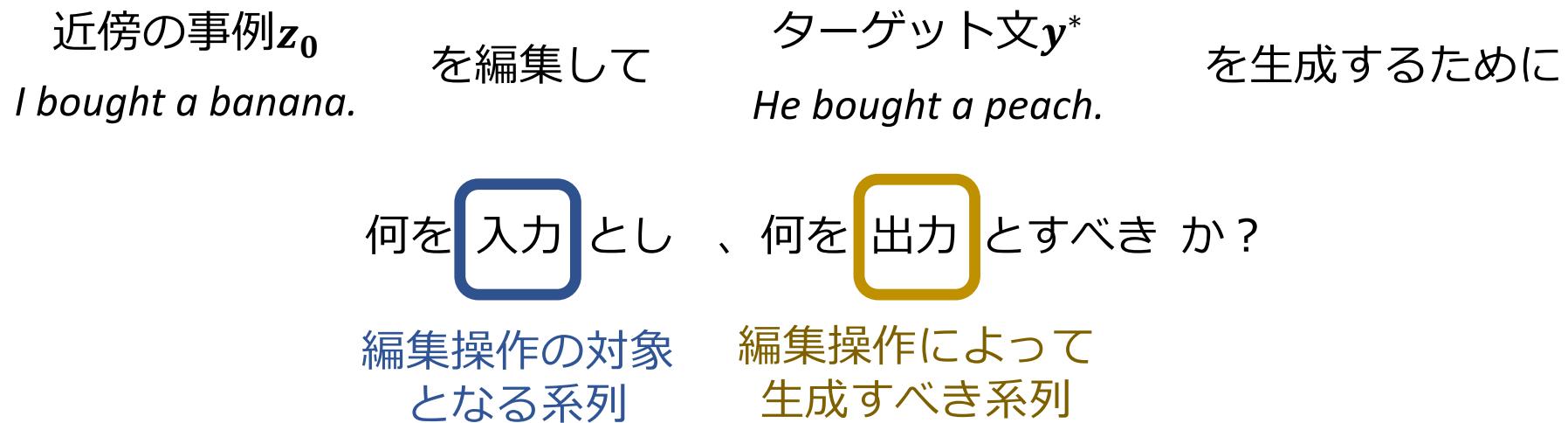
y^{k-1} : 前ステップまでに編集された系列

$k = 0$ では近傍の事例

近傍の事例を編集するための 方策分類器の学習

提案手法は模倣学習を用いて近傍の事例を編集するための
方策分類器 π を学習する。

理想的な編集操作を分類器に学習させるための事例の作成方法
(=オラクル方策 π^*) の設計がポイント

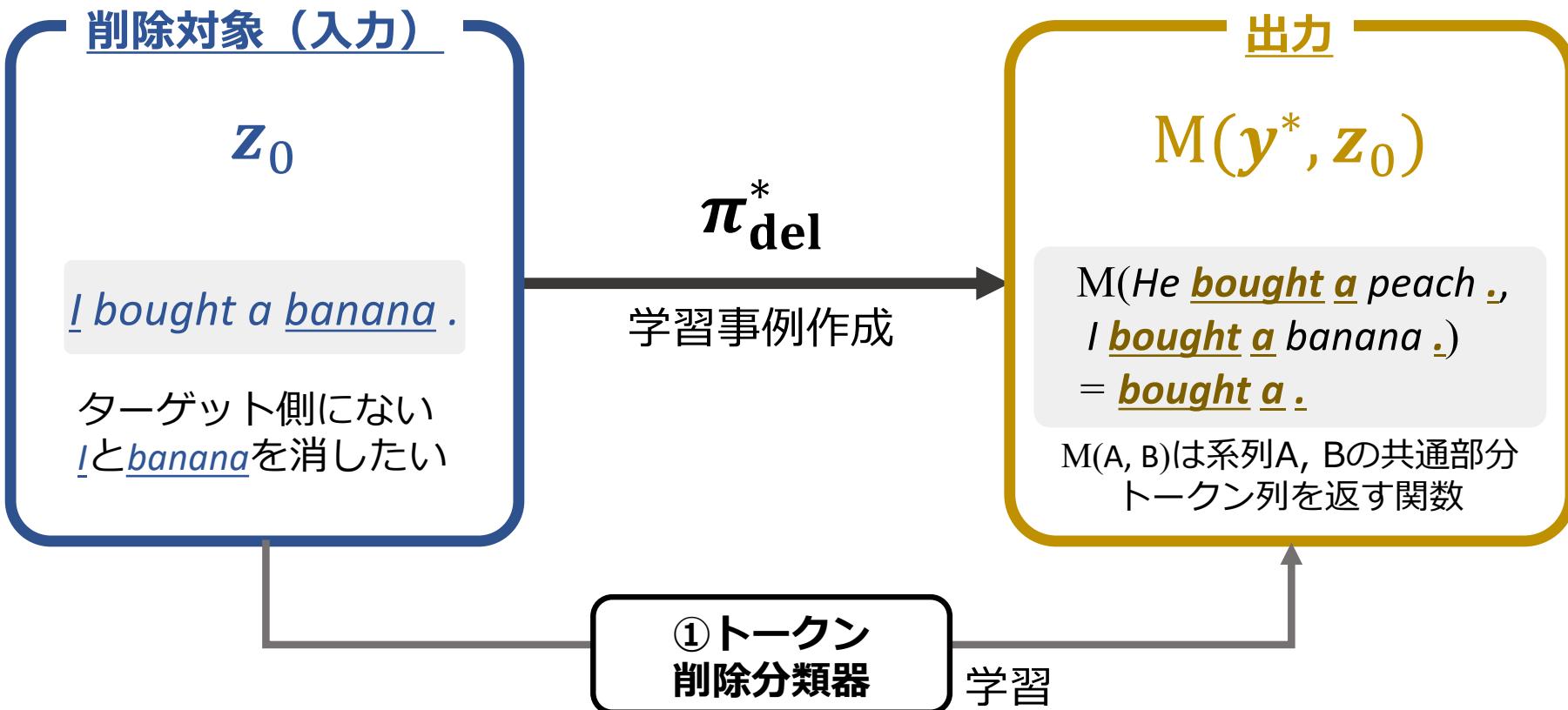


本研究では、近傍の事例に対する編集操作を効率的に学習させるため、
近傍の事例とターゲット文の差分に着目した学習方策を提案

削除操作のオラクル方策 π_{del}^*

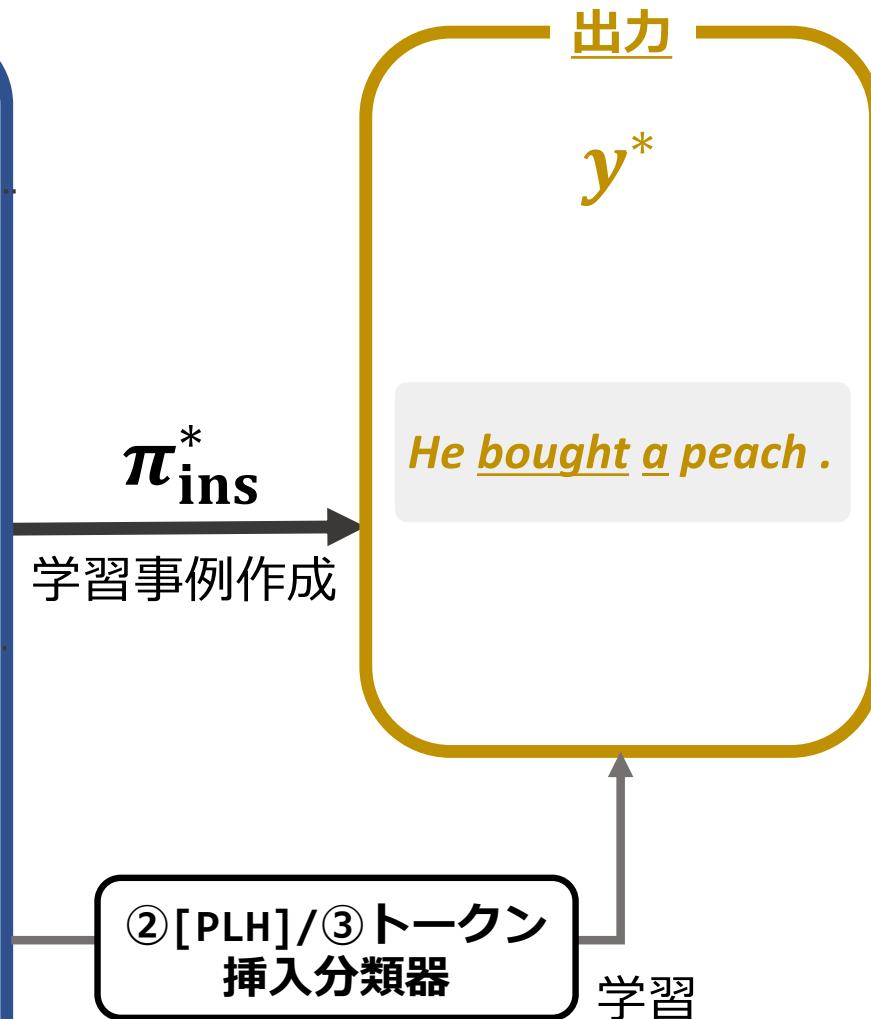
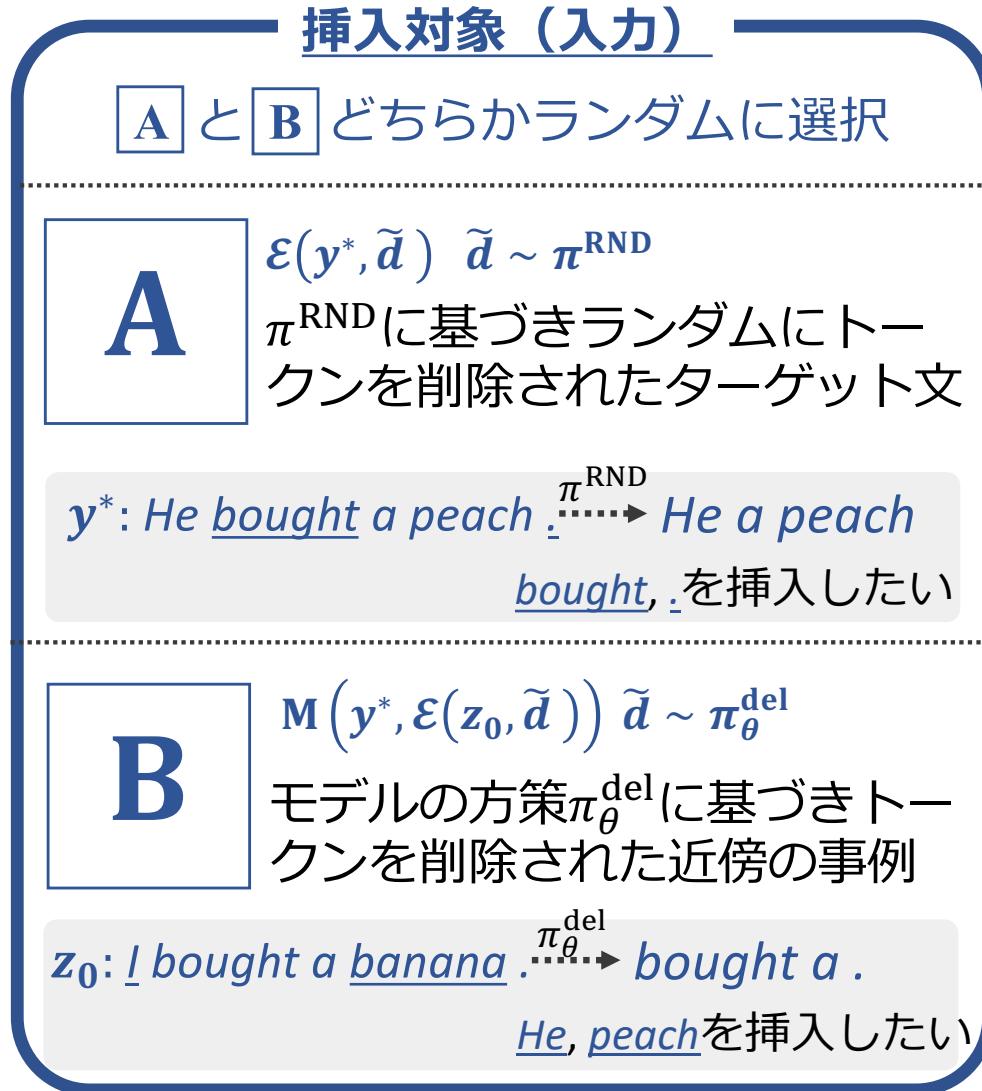
近傍の事例 z_0 を編集して ターゲット文 y^* を生成する
I bought a banana. *He bought a peach.*

[目的] 近傍の事例 z_0 から不要なトークンを削除する π_{del}^* の設計



挿入操作のオラクル方策 π_{ins}^*

[目的] 入力系列（編集済みの近傍の事例 z_0 ）をターゲット文 y^* に近づける



学習と推論

学習時

オラクル方策に基づき各分類器を学習

①トークン削除分類器

②[PLH]/③トークン挿入分類器

出力

オラクル方策で事例を作成

削除対象

挿入対象

推論時

終了条件を満たしたら終了
上限反復数など

y^k

③トークン挿入分類器

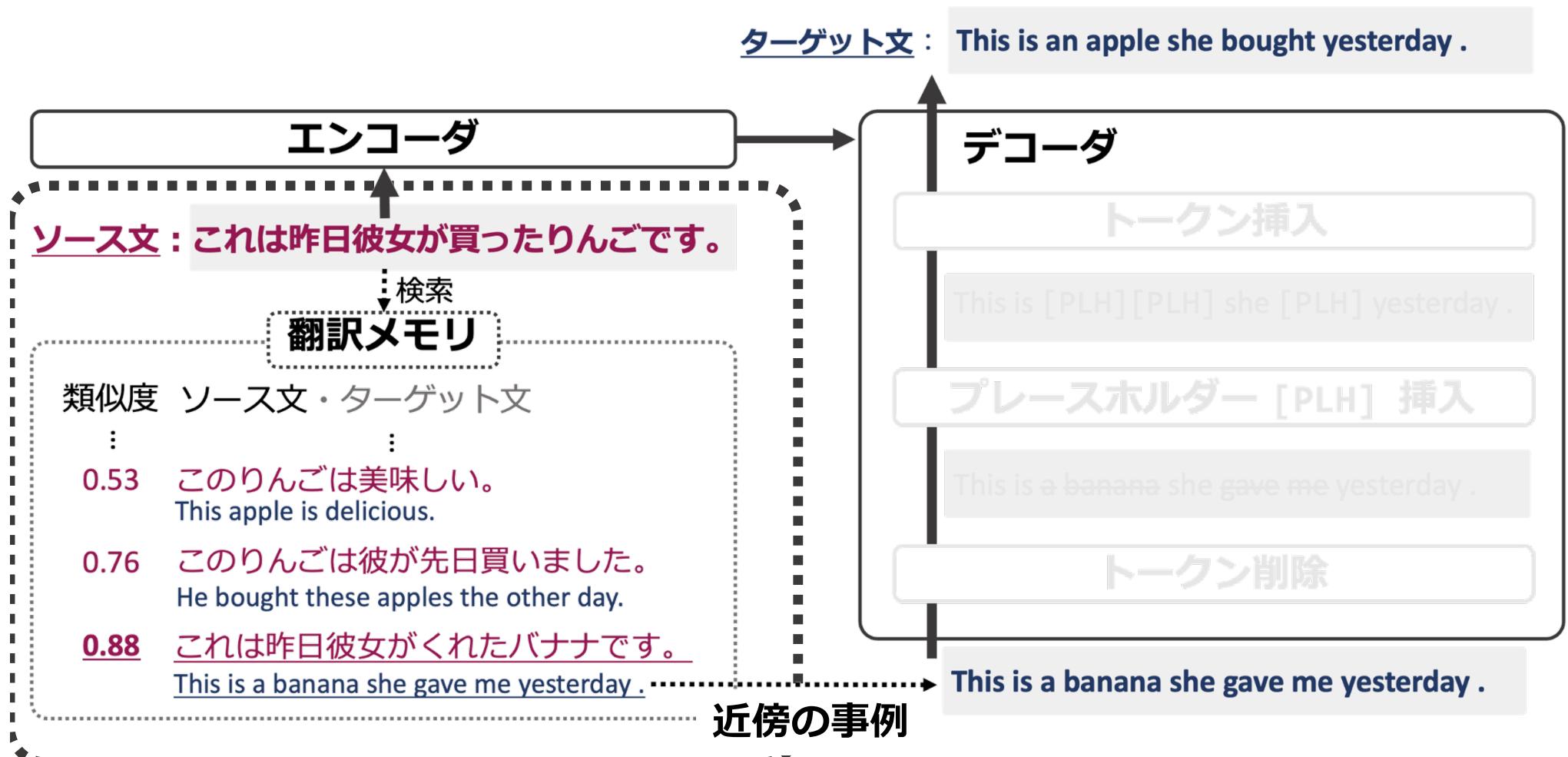
②[PLH]挿入分類器

①トークン削除分類器

y^{k-1}

提案手法: *NeighborEdit*

ソース文と近傍の事例を用いてターゲット文を生成する系列変換タスク



近傍の事例の検索

クエリ q

ソース文：これは昨日彼女が買ったりんごです。

翻訳メモリ

学習データ内の全ての文ペア（ソース文 s_i ・ ターゲット文 t_i ）が含まれるデータベース

s_1 : このりんごは美味しい。

t_1 : This apple is delicious .

クエリ q と翻訳メモリ
内の各ソース文 s_i との
近さスコア $S(q, s_i)$

$$S(q, s_1) = 0.53$$

s_2 : このりんごは彼が先日買いました。

t_2 : He bought these apples the other day .

$$S(q, s_2) = 0.76$$

s_3 : これは昨日彼女がくれたバナナです。

t_3 : This is a banana she gave me yesterday .

$$S(q, s_3) = 0.88$$

$S(q, s_i)$ が最大となるソース文を近傍探索ライブラリ faiss で検索し、
そのターゲット文を近傍の事例とする

近さスコア $S(q, s_i)$ の求め方

情報検索で従来から用いられているTF-IDFベクトルの
コサイン類似度 (**TFIDF**)

$$S_{\text{TFIDF}}(q, s_i) = \frac{\text{tfidf}(q) \cdot \text{tfidf}(s_i)}{\|\text{tfidf}(q)\| \|\text{tfidf}(s_i)\|}$$

tfidf(・): TF-IDFベクトル

[Option] より大域的な系列の類似度を考慮する**TFIDF+LCS**

$S_{\text{TFIDF}}(\cdot)$ の値が上位 n 件の事例を最長共通文字列 (LCS) の長さでランキング

例) ABCDEとAFDEGのLCSはADEで長さは3

BERTに基づき計算される文ベクトル h_q, h_{s_i} のコサイン類似度
(**SentVec**)

$$S_{\text{SentVec}}(q, s_i) = \frac{h_q \cdot h_{s_i}}{\|h_q\| \|h_{s_i}\|}$$

実験設定

データセット

法文書コーパス JRC-Acquis (本スライド内の結果は全て英独の翻訳タスク)

- 全ての文書が意味的に関連。文の意味的近さを利用する提案手法のベンチマークとして適している

評価指標 注目すべきは品質と速度

生成される文の品質: **BLEU**

生成する速度: 一文を生成するのに要する**平均反復数**および**平均時間** (ms)

※提案手法の平均時間には事例検索にかかる時間も含める

モデル	デコーダの初期値
Transformer (AR)	[BOS]
Levenshtein Transformer (NAR)	< s > </ s >
NeighborEdit (NAR)	近傍の事例 (・ランダムな事例) 検索方法: TFIDF, TFIDF+LCS, SentVec

本研究の概要

目的

非自己回帰モデルによる生成文の品質をより少ない生成回数で向上

アイデア

近傍の事例を非自己回帰モデルに組み込む

近傍の事例×非自己回帰モデルに取り組んだ既存研究はない

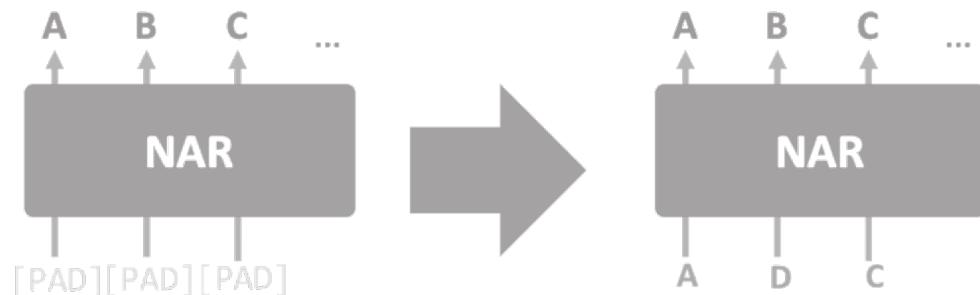
① デコーダの入力を近傍の事例で初期化

- Levenshtein Transformerをベースとし、近傍の事例を繰り返し編集して文を生成

② 近傍の事例に対する編集操作を効率的に学習するための学習方策を設計

することの有効性を調べたい

全てのトークンを無から生成するよりも容易に文を生成できる



近傍の事例をデコーダの初期値とする効果

モデル	デコーダの初期値	品質		速度	
		BLEU ↑	平均反復数 ↓	平均時間 (ms) ↓	速度
Transformer (AR)	[BOS]	54.25	27.10	313.1	
Levenshtein Trans. (NAR)	< s > < /s >	48.17	2.38	82.2	
NeighborEdit (NAR)	TFIDF	51.26	1.97	80.4	
	TFIDF+LCS	52.39	1.90	78.3	最大 -20%
	SentVec	52.37	1.95	80.4	
	ランダム	25.58 悪化	3.88 減少	131.0	

※太字は非自己回帰モデルでの最良値

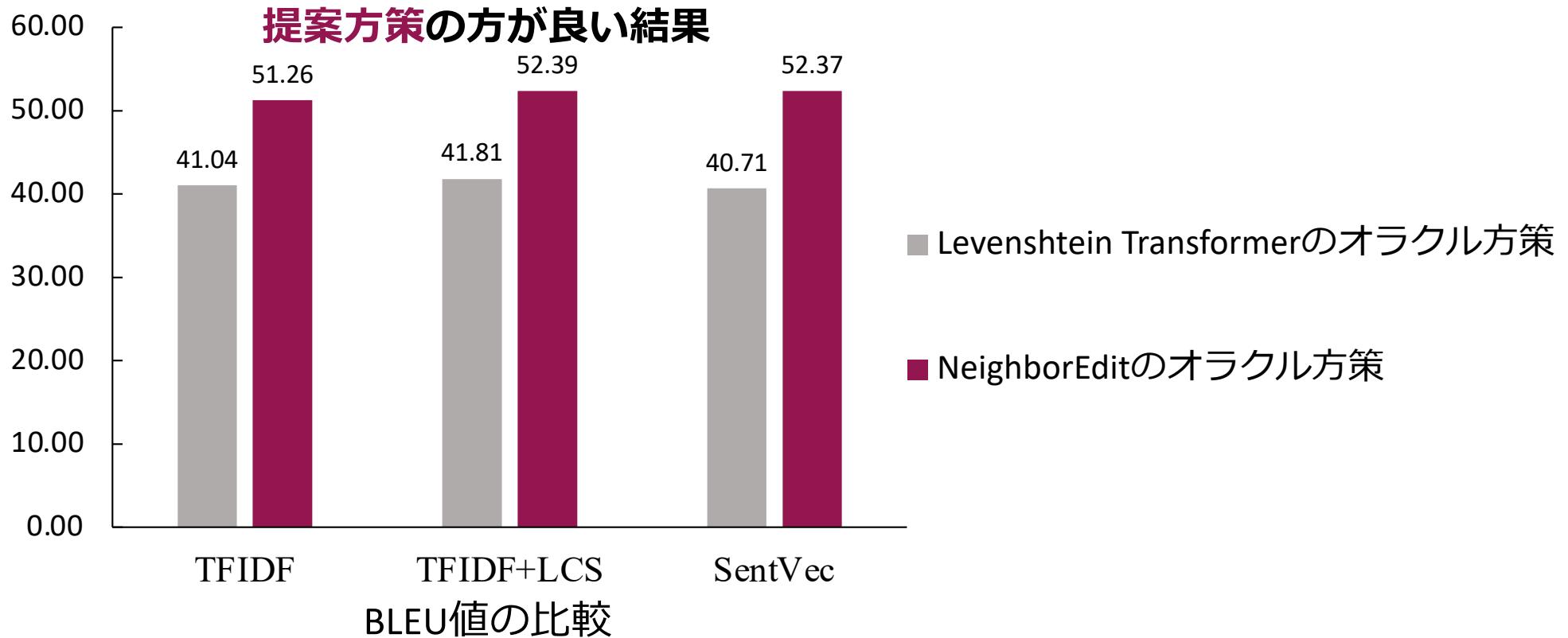
NARのベースラインと比較して **性能↑ 速度↓**

非自己回帰モデルのデコーダの初期値に近傍の事例を用いることは有効

推論時の検索時間は一件あたり2 (ms) 以下で十分短い

提案するオラクル方策の有効性

編集に基づく非自己回帰モデルのベースラインであるLevenshtein Transformerのオラクル方策で学習した結果と比較する



近傍の事例を編集するための提案方策も重要

近傍の事例をデコーダの初期値とする効果



モデル	デコーダの初期値	品質		速度	
		BLEU ↑	平均反復数 ↓	平均時間 (ms) ↓	速度
Transformer (AR)	[BOS]	54.25 最高値	27.10	313.1	
Levenshtein Trans. (NAR)	< s > </ s >	48.17	2.38	82.2	
NeighborEdit (NAR)	TFIDF	51.26	1.97	80.4	
	TFIDF+LCS	52.39	1.90	78.3	
	SentVec	52.37	1.95	80.4	
	ランダム	25.58	3.88	131.0	

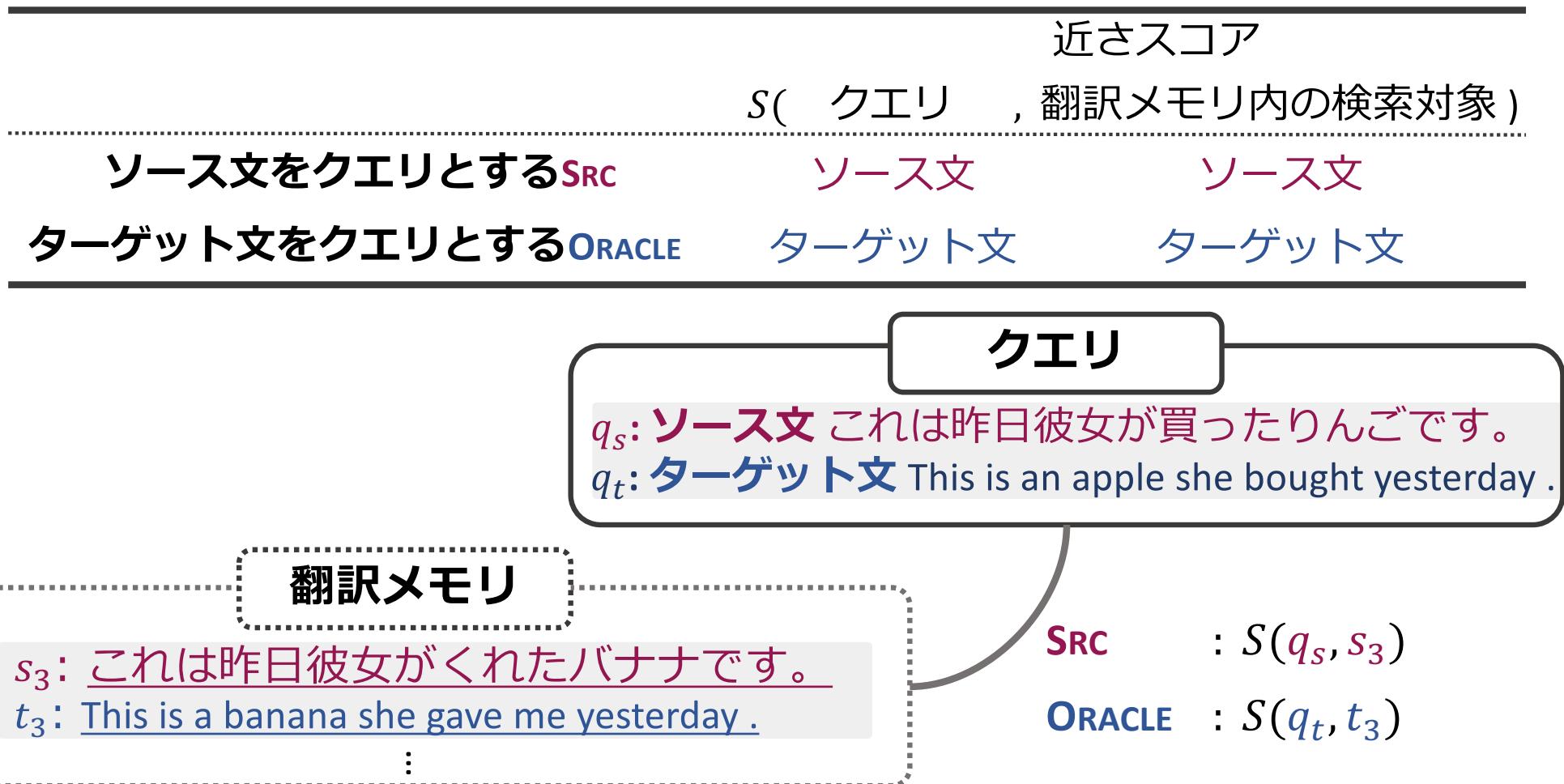
※太字は非自己回帰モデルでの最良値

しかしながら、自己回帰モデル (AR) のベースラインと比較して文の品質は劣る

提案手法の性能をさらに向上させるためには？近傍の事例の質が性能に与える影響を調査

近傍の事例が性能に与える影響

学習時と推論時の検索クエリにそれぞれソース文とターゲット文を用いて比較



ORACLEはターゲット文に一番近い文を近傍の事例とするため、
理想的な近傍の事例を選択できた場合の性能と考えられる 20

近傍の事例が性能に与える影響

		推論時のクエリ		
		学習時の クエリ	SRC	ORACLE
TFIDF	SRC	51.26	52.95	SRC: ソース文同士が最も近い近傍の事例
	ORACLE	52.69	56.51*	
TFIDF +LCS	SRC	52.39	54.92*	ORACLE: ターゲット文同士が最も近い近傍の事例
	ORACLE	52.93	57.09*	
SentVec	SRC	52.37	52.76	*のある数値: ARモデルの性能を超えた数値
	ORACLE	52.54	53.51	

適切な近傍の事例を選択できれば、提案手法は自己回帰モデル(AR)の性能を超える可能性。しかし実際にはORACLEは使えない...

SRCでORACLEの性能に迫れるか？

上位1件ではなく上位5件の近傍の事例を用いて並列に文を生成し、最もBLEUの値が高い事例を出力文と見なす (Top-5)

		推論時のクエリ					
		学習時の クエリ		SRC		ORACLE	
		SRC		Top-1	Top-5	Top-1	Top-5
TFIDF	SRC	51.26	<	56.09*		52.95	56.63*
	ORACLE	52.69				56.51*	
TFIDF +LCS	SRC	52.39	<	57.33*		54.92*	58.20*
	ORACLE	52.93				57.09*	
SentVec	SRC	52.37	<	57.76*		52.76	57.50*
	ORACLE	52.54				53.51	

SRC:
ソース文同士が最も近い近傍の事例

ORACLE:
ターゲット文同士が最も近い近傍の事例

*のある数値:
ARモデルの性能を超えた数値

- SRCでも4.8ポイント以上向上。複数の近傍の事例を用いて文を生成しランキングすることで大幅な性能向上が見込める
- どうやって質の高い（理想的な）近傍の事例を持ってくるか・複数の近傍の事例を考慮するかが今後の課題

本研究のまとめ

目的

非自己回帰モデルによる生成文の品質をより少ない回数の生成で向上

提案: 非自己回帰モデルに近傍の事例を組み込む

デコーダの最初の入力を近傍の事例で初期化。近傍の事例とターゲット文の差分に着目した学習方策により、近傍の事例を編集して文を生成。

実験からわかったこと

- ・ 非自己回帰モデルにおいても近傍の事例は有効
- ・ 理想的な近傍の事例が検索できれば自己回帰モデルの性能を超える可能性

今後の予定

- ・ 距離の遠い近傍の事例に対しても有効なオラクル方策の検討
- ・ 複数の事例を考慮する手法の開発や"理想的な"近傍の事例に関する調査