文脈内学習に基づくテキストスタイル変換における例示組合せ最適化

大庭　知也† 渡邊　千紘†

† NTT コンピュータ&データサイエンス研究所　〒 180–8585 東京都武蔵野市緑町 3-9-11
E-mail: †{tomoya.ohba,ch.watanabe}@ntt.com

あらまし　大規模言語モデルの発展に伴い，事前学習済みの大規模言語モデルを用いてテキストスタイル変換を行うアプローチが提案されている．このようなアプローチにおいて，文脈内学習の枠組みに基づき例示を含むプロンプトを入力とする場合，例示組合せの選択がタスク性能に影響を与えることが指摘されている．テキストスタイル変換においては，単一のスタイルを持つテキストからなる非パラレルデータと，異なるスタイルを持つテキストの組からなるパラレルデータの 2 種類の形式を例示に含めることが可能である．しかし，例示に含める際に，これらの形式の違いがタスク性能に及ぼす影響は明らかになっていない．そこで，本研究では，パラレル・非パラレル両データを含みうるデータセットから最適な例示組合せを選択するアルゴリズムを提案し，異なる形式のデータセットに適用することで，パラレルデータにおける変換関係の情報の有無がテキストスタイル変換のタスク性能に与える影響を検証する．実験の結果，本研究における設定の下では，高精度なテキストスタイル変換を実現するために変換関係の情報が重要であることが示された．

キーワード　テキストスタイル変換，文脈内学習，プロンプト最適化，大規模言語モデル

1 はじめに

近年，大規模言語モデル（Large Language Model; LLM）の発展に伴い，質問応答 [22] や文章要約 [32]，機械翻訳 [13] など，様々な応用において高精度なテキスト処理が可能となっている [15]．本研究では，特にテキストスタイル変換（Text Style Transfer; TST）タスクに着目する．これは，テキストの（スタイルに非依存な）内容を保持したままスタイルのみを変換するタスクであり，チャットボットによる対話システム [16] やテキスト作成支援システム [1] [14] などへの応用が可能である．

TST を実現する主なアプローチとして，TST タスクに特化したモデルを設計し学習する方法と，事前学習済み LLM による推論に基づく方法が存在する．前者のアプローチにおいては，モデルの TST タスクに関する性能を直接最適化することが可能である一方，学習の計算コストが高いという問題がある [26]．後者のアプローチでは，事前学習済みの汎用 LLM を追加学習なしで用いることにより，推論時の計算コストのみで TST を実現することができる．また，パラメータ情報にアクセスできないブラックボックスなモデルを活用できるという利点があり，近年多くの手法が提案されている [27]．

上記のように，事前学習済み LLM を用いて与えられたタスクを解くアプローチにおいて，高精度なタスク性能を達成するための方法として，自動プロンプト最適化（Automatic Prompt Optimization; APO）が提案されている [34]．これは，LLM に入力するプロンプトをタスクに合わせて最適化することにより，モデルパラメータの追加学習なしでタスク精度を向上することを目的とした手法である．特に，タスク記述を含む指示，タスクの問題文とその回答例を記述する例示，クエリからプロンプトを構成する文脈内学習（In-Context Learning; ICL）[4] の枠組みにおいて，例示を最適化することの重要性が近年指摘されている [28]．

(a) 非パラレルデータ

スタイル	テキスト
positive	あの映画は素晴らしかった.
positive	ここのレストランの料理は美味しい.
negative	この部屋は狭くて汚い.
⋮	⋮

(b) パラレルデータ

ソース スタイル	ソーステキスト	ターゲット スタイル	ターゲットテキスト
positive	あの映画は素晴らしかった.	negative	あの映画はひどかった.
positive	ここのレストランの料理は美味しい.	negative	ここのレストランの料理はまずい.
negative	この部屋は狭くて汚い.	positive	この部屋は広くて綺麗だ.
⋮	⋮	⋮	⋮

図 1　(a) 非パラレルデータと (b) パラレルデータの例.

TST タスクにおいてプロンプトの例示最適化を行う際，タスク固有の特徴として，単一のテキストからなる非パラレルデータと，スタイル変換前後のテキストの組からなるパラレルデータの 2 種類の形式が存在する（図 1）[5] [9] [10]．パラレルデータは非パラレルデータと比べ，スタイル変換前後のテキストを比較可能な形の情報を含むため，例示として用いることでより高精度な TST を達成できることが期待される一方，データ収集のコストが高い．そのため，高精度な TST を実現するためにパラレルデータ形式の例示が必要であるのか，もしくは非パラレルデータ形式の例示のみで同程度のタスク性能を実現できるのかについて検証することは重要な研究課題である．このような検証を行うためには，パラレルデータにおける変換関係の情報を保持する設定と保持しない設定の両ケースについて，統一的な枠組みに基づき例示選択を行うことが可能な手法を構築し，それらの結果を比較する必要がある（図 2）．

そこで，本研究では，パラレル・非パラレル両形式のデータ

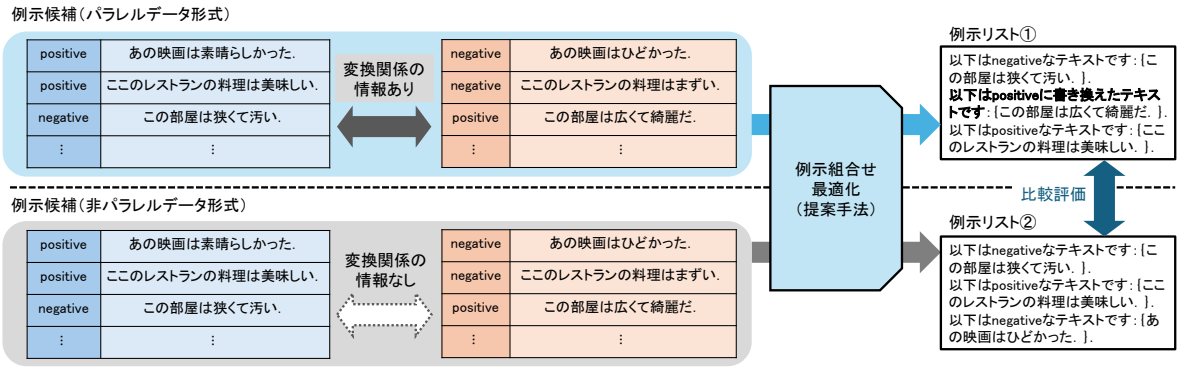

図 2　パラレルデータにおける変換関係の情報を保持する設定と保持しない設定における例示選択結果の比較評価の枠組み.

セットに関して得られる例示選択の結果を統一的に評価する枠組みを提案する．ここで，形式上パラレルデータは 2 つの，非パラレルデータは 1 つのテキストを含むことに注意する．そのため，それぞれを例示の 1 サンプルとして扱った場合，変換関係の情報の有無に加え，例示テキストの長さの違いがタスク性能に影響する可能性がある．したがって，提案法ではパラレルデータにおける変換前後のテキストをそれぞれ個別のサンプルとして扱ったうえで，その変換関係の情報に基づき貪欲法による例示最適化を行う．さらに，提案手法をパラレルデータあり・なしのデータセットに適用し，両ケースで選択された例示組合せのテスト性能を比較することにより，パラレルデータにおける変換関係の情報の有無が TST のタスク性能に与える影響について検証を行う．

2　関 連 研 究

2.1　テキストスタイル変換

TST を実現するためのアプローチとして，従来，TST タスクに特化した独自のモデルを設計し，学習する方法が提案されてきた．特に，1 節に述べた通り，異なるスタイルを持つテキストの組を含むパラレルデータは収集コストが高いため，非パラレルなデータから教師なし学習に基づく TST が可能なエンコーダ・デコーダ型のモデルが提案されている [6] [12] [23]. TST において，変換前（後）のスタイルとテキストをそれぞれソース（ターゲット）スタイル，ソース（ターゲット）テキストと呼ぶ．上記のモデルでは，まずエンコーダにおいて入力テキストのスタイル情報とスタイルに非依存な情報を分離した上で，デコーダにおいてターゲットテキストへの変換を行う．このアプローチにおいては，TST タスクに関する性能を直接最適化したモデルを獲得できる一方，学習時に高い計算コストがかかるという問題がある．

近年，上記の学習時における計算コストの問題を回避することが可能なアプローチとして，事前学習済み LLM を用いて TST を実現する手法が提案されている [17] [20] [24]．これらはデコーダ型の事前学習済み LLM に対し，ソーステキストのターゲットスタイルへの変換を指示するプロンプトを入力することで，ターゲットテキストの予測を行わせる方法である．これにより，推論時の計算コストのみで TST を実現でき，ブラックボックスなモデルも活用可能であるという利点がある．このようなアプローチにおいては，LLM に入力するプロンプトの設定により TST の性能が変わりうることが知られており，特に ICL の枠組みに基づき適切な例示の組合せをプロンプトに含めることで，タスク性能が向上することが報告されている [20].

2.2　自動プロンプト最適化

2.1 節に述べた通り，事前学習済み LLM の推論を通して TST を実現する手法においては，LLM に入力するプロンプトを適切に定義することが重要となる．一般に，事前学習済み LLM を用いて与えられたタスクを解くアプローチにおいて，タスク性能を最大化するプロンプトを推定する APO の手法が提案されている [2] [19] [31] [34]．特に，APO の性能を検証することを目的として，TST タスクに手法を適用した例も存在している [3] [8] [29]．ただし，これらの研究においては，プロンプトにおける例示組合せの最適化を実現する手法は提案されていない.

近年，ICL の枠組みに基づき指示と例示からなるプロンプト構成を想定する場合において，例示を最適化することの重要性が指摘されている [28]．一般のタスクに対し，プロンプト内で提示する例示組合せの最適化を行う手法はすでに提案されているが [25] [30]，これらの研究においては TST タスク固有の特徴であるパラレル・非パラレル両データの存在を考慮した手法は提案されていない.

2.3　本研究の位置づけ

2.1 節に述べたように，近年，事前学習済み汎用 LLM を用いて TST タスクを解くアプローチが提案されている．一方で，2.2 節に述べたように，LLM を用いて一般のタスクを解く際の性能向上を目的として APO が提案されている．特に，プロンプトに含まれる例示組合せを適切に設定することにより，TST のタスク性能が向上することが期待されるが，TST タスク固有の特徴であるパラレル・非パラレル両データの存在を考慮した上で最適な組合せを推定する手法は提案されていない.

本研究では，パラレル・非パラレル両データを含みうるデータセットから TST タスクにおいて最適な例示組合せを推定す

る手法を提案する．提案手法は，事前学習済み LLM を用いて TST タスクを解く設定における APO の手法の一種であり，提案手法により選択された例示組合せをプロンプトに含めることでタスク性能の向上を見込むことができる．

3 提案手法

本研究では，ICL の枠組みに基づき事前学習済み LLM を用いて TST タスクを解く設定において，プロンプト内で提示する例示組合せの最適化を行うアルゴリズムを提案する．提案手法の概要を図 3 に示す．まず，提案手法で用いるデータセットについての説明を述べた後（3.1 節），提案手法のアルゴリズムについて説明する（3.2 節）．

3.1 データセット

本研究では，パラレル・非パラレル両データが含まれうるデータセットの中から TST タスクにおいて最適な例示組合せを選択する問題を考える．与えられた TST タスクにおいて扱うスタイルの集合を $\mathcal{S} = \{s_1, \ldots, s_K\}$ とする（ただし，K をスタイルの数，$\mathcal{T}$ を存在しうるすべてのテキストの集合とし，$i = 1, \ldots, K$ について $s_i \in \mathcal{T}$ とする）．本稿においては，スタイルとして肯定的（positive），否定的（negative）の 2 種類のみを考える（$\mathcal{S} = \{\text{“positive”}, \text{“negative”}\}$）．TST とは，あるスタイル $s \in \mathcal{S}$ を持つテキスト $t \in \mathcal{T}$ とターゲットスタイル $\tilde{s} \in \mathcal{S}$ の組合せが与えられたとき，テキスト t が持つスタイルに非依存な内容を保持したままスタイル $\tilde{s}$ に変換するタスクとして定義することができる．

本研究では，TST のデータセットとして，以下の 2 種類の形式を想定する．

- **非パラレルデータセット**：非パラレルデータセット $\mathcal{D}_1 = \{(s_1, t_1), \ldots, (s_{n_1}, t_{n_1})\}$ は，スタイル $s \in \mathcal{S}$ と，スタイル s を持つテキスト $t \in \mathcal{T}$ の組 (s, t) の集合で表される．ただし，n_1 はサンプルサイズを表す．
- **パラレルデータセット**：パラレルデータセット $\mathcal{D}_2 = \{(s_1, t_1, \tilde{s}_1, \tilde{t}_1), \ldots, (s_{n_2}, t_{n_2}, \tilde{s}_{n_2}, \tilde{t}_{n_2})\}$ は，ソーススタイル $s \in \mathcal{S}$ と，スタイル s を持つソーステキスト $t \in \mathcal{T}$ と，ターゲットスタイル $\tilde{s} \in \mathcal{S}$ と，テキスト t をスタイル $\tilde{s}$ に変換したターゲットテキスト $\tilde{t} \in \mathcal{T}$ の組 $(s, t, \tilde{s}, \tilde{t})$ の集合で表される．ただし，n_2 はサンプルサイズを表す．

上記の形式で与えられる非パラレルデータセット $\mathcal{D}_1$ とパラレルデータセット $\mathcal{D}_2$ に基づき [1]，例示組合せの最適化を行うために，まずこれらのデータセットを学習データセット $\mathcal{D}_{\text{train}} \subset \mathcal{D}_2$, テストデータセット $\mathcal{D}_{\text{test}} \subset \mathcal{D}_2$, 例示データセット $\mathcal{D}_{\text{exemplar}} \subset \mathcal{D}_1 \cup \mathcal{D}_2$ に分割する．ただし，いずれのデータセットも少なくとも 1 つの要素を含み，かついずれの異なる 2 つのデータセットの組も互いに素であるように定義しておく．

さらに，例示データセット $\mathcal{D}_{\text{exemplar}}$ のうち非パラレルデータの集合を $\mathcal{D}_{\text{NP}} = \{(s_1^{\text{NP}}, t_1^{\text{NP}}), \ldots, (s_{n_{\text{NP}}}^{\text{NP}}, t_{n_{\text{NP}}}^{\text{NP}})\}$, パラレルデータの集合を $\mathcal{D}_{\text{P}} = \{(s_1^{\text{P}}, t_1^{\text{P}}, \tilde{s}_1^{\text{P}}, \tilde{t}_1^{\text{P}}), \ldots, (s_{n_{\text{P}}}^{\text{P}}, t_{n_{\text{P}}}^{\text{P}}, \tilde{s}_{n_{\text{P}}}^{\text{P}}, \tilde{t}_{n_{\text{P}}}^{\text{P}})\}$ とし（$\mathcal{D}_{\text{NP}} \subset \mathcal{D}_1$, $\mathcal{D}_{\text{P}} \subset \mathcal{D}_2$, $\mathcal{D}_{\text{exemplar}} = \mathcal{D}_{\text{NP}} \cup \mathcal{D}_{\text{P}}$），これらに基づき以下の例示集合 $\mathcal{X}_1$, $\mathcal{X}_2$, $\mathcal{X}_3$ を定義する．ただし，$(s_i^{\text{P}}, t_i^{\text{P}}, \tilde{s}_i^{\text{P}}, \tilde{t}_i^{\text{P}})$ を $\mathcal{D}_{\text{P}}$ の i 番目の要素とし，$(s_i^{\text{NP}}, t_i^{\text{NP}})$ を $\mathcal{D}_{\text{NP}}$ の i 番目の要素とする．まず，ソーステキストからなる例示の集合として $\mathcal{X}_1$ を定義する．これは，スタイルに非依存な内容が同一であるようなテキストの連なりにおいて，1 番目に提示されるテキストの集合とも言い換えることができる．

$$\mathcal{X}_1 = \{x_1^{(1)}, \ldots, x_{m1}^{(1)}\} = u(\{\tilde{x}_1^{(1)}, \ldots, \tilde{x}_{n_{\text{P}}+n_{\text{NP}}}^{(1)}\}),$$

$$\tilde{x}_i^{(1)} = \begin{cases} h_1(s_i^{\text{P}}, t_i^{\text{P}}), \text{for } i = 1, \ldots, n_{\text{P}}, \\ h_1(s_{i-n_{\text{P}}}^{\text{NP}}, t_{i-n_{\text{P}}}^{\text{NP}}), \text{for } i = n_{\text{P}}+1, \ldots, n_{\text{P}}+n_{\text{NP}}, \end{cases}$$

$$h_1(s, t) = \text{“Here is a text, which is } s\text{: } \{t\}. \text{”} \quad (1)$$

ここで，u は入力のテキスト集合から重複を除いた集合を出力する関数とする．次に，ターゲットテキストからなる例示の集合として $\mathcal{X}_2$ を定義する．これは，上記のテキストの連なりにおいて，2 番目に提示されるテキストの集合となる．

$$\mathcal{X}_2 = \{x_1^{(2)}, \ldots, x_{m2}^{(2)}\} = u(\{\tilde{x}_1^{(2)}, \ldots, \tilde{x}_{n_{\text{P}}}^{(2)}\}),$$

$$\tilde{x}_i^{(2)} = h_2(\tilde{s}_i^{\text{P}}, \tilde{t}_i^{\text{P}}), \text{for } i = 1, \ldots, n_{\text{P}},$$

$$h_2(s, t) =$$

$$\text{“Here is a rewrite of the text, which is } s\text{: } \{t\}. \text{”} \quad (2)$$

さらに，ターゲットテキストからなる新たな例示の集合として $\mathcal{X}_3$ を定義する．これは，上記のテキストの連なりにおいて，3 番目以降に提示されるテキストの集合となる．

$$\mathcal{X}_3 = \{x_1^{(3)}, \ldots, x_{m3}^{(3)}\} = u(\{\tilde{x}_1^{(3)}, \ldots, \tilde{x}_{n_{\text{P}}}^{(3)}\}),$$

$$\tilde{x}_i^{(3)} = h_3(\tilde{s}_i^{\text{P}}, \tilde{t}_i^{\text{P}}), \text{ for } i = 1, \ldots, n_{\text{P}},$$

$$h_3(s, t) =$$

$$\text{“Here is another rewrite of the text, which is } s\text{: } \{t\}. \text{”} \quad (3)$$

最後に，$\mathcal{X}_1, \mathcal{X}_2, \mathcal{X}_3$ の要素同士の変換関係の情報を参照可能にしておくため，これらの集合の各要素に対応するソーステキストの番号を出力する関数 $\phi : \mathcal{T} \mapsto \mathbb{N}$ を以下のように定義する．まず，$\mathcal{X}_1$ の各要素 $x_i^{(1)}$ $(i = 1, \ldots, m_1)$ について，$\phi(x_i^{(1)}) = i$ とする．次に，$\mathcal{X}_2$ の各要素 $x_i^{(2)}$ $(i = 1, \ldots, m_2)$ について，対応するソーステキストが含まれる $\mathcal{X}_1$ の要素が $x_j^{(1)}$ であるとき，$\phi(x_i^{(2)}) = j$ とする．さらに，$\mathcal{X}_3$ の各要素 $x_i^{(3)}$ $(i = 1, \ldots, m_3)$ について，対応するソーステキストが含まれる $\mathcal{X}_1$ の要素が $x_j^{(1)}$ であるとき，$\phi(x_i^{(3)}) = j$ とする．

3.2 貪欲法に基づく例示組合せ最適化

本研究の目的は，3.1 節で定義した学習データセット $\mathcal{D}_{\text{train}}$ について，TST のタスク性能を最大化する例示組合せを例示集合 $\mathcal{X}_1, \mathcal{X}_2, \mathcal{X}_3$ から選択することである．すべての可能な例示組合せの中から学習損失を最小化する組合せを選択することは計算的に困難であるため，本研究では貪欲法に基づく例示組合せ最適化手法を提案する（アルゴリズム 1）．これは，具

1：ここで，非パラレルデータセット $\mathcal{D}_1$ については空集合でもよい．

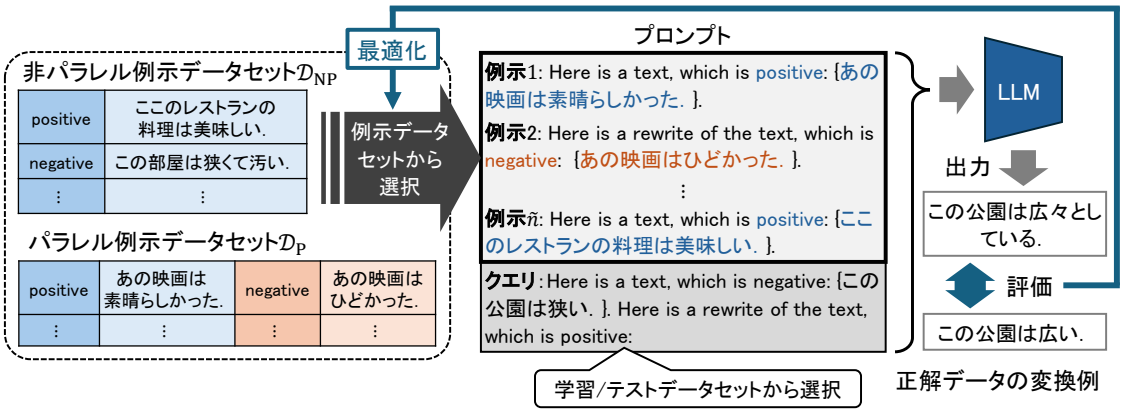

図 3　文脈内学習に基づくテキストスタイル変換における例示組合せ最適化の枠組み.

体的には例示数 $\tilde{n} \in \mathbb{N}$ が与えられたとき，学習データセット $\mathcal{D}_{\mathrm{train}}$ における損失を最小化するようなサイズ $\tilde{n}$ の例示リスト $\hat{\mathcal{Y}} = [y_1, \ldots, y_{\tilde{n}}]$ を選択する手法である．

貪欲法に基づくナイーブな例示選択方法として，1 番目の例示から順に学習損失を最小化する候補を選択していく方法が考えられるが，この方法では例示の選択順が固定されているため，局所最適解に陥りやすい．そのため，本研究では，例示の選択順をランダムに変えて T 回の例示組合せ選択を行い，それらの結果のうち学習損失を最小化する組合せを例示リスト $\hat{\mathcal{Y}}$ として出力することを提案する．ただし，T は任意の自然数とする．

提案手法においては，3.1 節に述べたスタイルに非依存な内容が同一であるようなテキストの連なりを表す例示リスト $\boldsymbol{g}_1, \ldots, \boldsymbol{g}_C$ を要素とする例示グループリスト $\mathcal{G} = [\boldsymbol{g}_1, \ldots, \boldsymbol{g}_C]$ を扱う．ただし，C を $\mathcal{G}$ の要素数とする．これにより，変換関係の矛盾が生じないように，位置に応じた例示集合の中から候補を選択することが可能になる．具体的には，各ステップ $i \in \{1, \ldots, T\}$ において，例示グループリスト $\mathcal{G}^{(i)}$ の初期値を空リストとする．各 $j \in \{1, \ldots, \tilde{n}\}$ について，一様分布から位置 r を生成し，現在の例示グループリスト $\mathcal{G}^{(i)}$ における位置 r に新たな例示を追加することを考える．ここで，位置 r に応じて，変換関係の矛盾なく追加することが可能な例示候補の集合として $\mathcal{X}$ を定義し（アルゴリズム 1 の 6–16 行目），その要素から最適な例示候補の選択を行う．

最適な例示候補の選択にあたり，各候補を現在の例示グループリスト $\mathcal{G}^{(i)}$ に追加した結果得られる例示リストについて，それを用いた場合の学習損失を計算する必要がある．このため，以下の関数 α, ζ を定義しておく．まず，例示グループリスト $\mathcal{G}$ の位置 $i \in \{1, \ldots, C+1\}$ に新たな例示 x を追加する操作を表す関数 α を以下のように定義する．(1) $i = 1$ の場合，$\alpha(\mathcal{G}, i, x) = [[x], \boldsymbol{g}_1, \ldots, \boldsymbol{g}_C]$．(2) $i \neq 1$ の場合，$\boldsymbol{g}_{i-1} = [x_1, \ldots, x_m]$ とする．(2a) $\phi(x) = \phi(x_1)$ かつ $x \neq x_1$ の場合，$\alpha(\mathcal{G}, i, x) = [\boldsymbol{g}_1, \ldots, \boldsymbol{g}_{i-2}, [x_1, \ldots, x_m, x], \boldsymbol{g}_i, \ldots, \boldsymbol{g}_C]$．(2b) $\phi(x) \neq \phi(x_1)$ もしくは $x = x_1$ の場合，$\alpha(\mathcal{G}, i, x) = [\boldsymbol{g}_1, \ldots, \boldsymbol{g}_{i-1}, [x], \boldsymbol{g}_i, \ldots, \boldsymbol{g}_C]$．次に，与えられた例示グループリスト $\mathcal{G}$ の各要素に含まれる例示を昇順に並べることで，例示リストに変換する関数 ζ を定義する．これは，$i = 1, \ldots, C$ について $\boldsymbol{g}_i = [x_1^{(i)}, \ldots, x_{m_i}^{(i)}]$ とするとき，以下のように定義される．

$$\zeta(\mathcal{G}) = [x_1^{(1)}, \ldots, x_{m_1}^{(1)}, \ldots, x_1^{(C)}, \ldots, x_{m_C}^{(C)}] \tag{4}$$

これらの関数に基づき，現在の例示グループリスト $\mathcal{G}^{(i)}$ の位置 r に j 個目の例示 x を追加することで得られる例示リストを $\mathcal{Y} = [y_1, \ldots, y_j] = \zeta(\alpha(\mathcal{G}^{(i)}, r, x))$ と表すことができる．

上記の例示リスト $\mathcal{Y}$ に対応する学習損失を計算するため，与えられた学習サンプル $\boldsymbol{q} = (s_{\mathrm{Q}}, t_{\mathrm{Q}}, \tilde{s}_{\mathrm{Q}}, \tilde{t}_{\mathrm{Q}}) \in \mathcal{D}_{\mathrm{train}}$ と例示リスト $\mathcal{Y}$ からプロンプト $p \in \mathcal{T}$ を出力する関数 ρ を定義する．一般的に，LLM に対するプロンプトは指示，例示，クエリの 3 点から構成されるが，本稿では簡単のため，例示とクエリのみに基づく以下の定義を用いる．

$$\begin{aligned} p = \rho(\boldsymbol{q}, \mathcal{Y}) &= y_1 + \cdots + y_j + h_{\mathrm{Q}}(s_{\mathrm{Q}}, t_{\mathrm{Q}}, \tilde{s}_{\mathrm{Q}}), \\ h_{\mathrm{Q}}(s, t, \tilde{s}) &= \text{“Here is some } s \text{ text: } \{t\}. \\ &\quad \text{Here is a rewrite of the text, which is } \tilde{s}\text{: } \{\text{”} \end{aligned} \tag{5}$$

ここで，$+$ は 2 つのテキストを結合し 1 つのテキストとする操作を表す．学習データセットを

$$\begin{aligned} \mathcal{D}_{\mathrm{train}} &= \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_{|\mathcal{D}_{\mathrm{train}}|}\}, \\ \boldsymbol{x}_i &= (s_i, t_i, \tilde{s}_i, \tilde{t}_i) \text{ for } i = 1, \ldots, |\mathcal{D}_{\mathrm{train}}| \end{aligned} \tag{6}$$

としたとき，与えられた例示リスト $\mathcal{Y}$ の学習損失は以下で定義される．

$$\mathcal{L}_{\mathrm{train}}(\mathcal{Y}) = \frac{1}{|\mathcal{D}_{\mathrm{train}}|} \sum_{i=1}^{|\mathcal{D}_{\mathrm{train}}|} d(\tilde{t}_i, f_{\mathrm{LLM}}(\rho(\boldsymbol{x}_i, \mathcal{Y}))) \tag{7}$$

ここで，$f_{\mathrm{LLM}} : \mathcal{T} \mapsto \mathcal{T}$ はテキストを入力とし，テキストを出力する任意の言語モデルを表す関数とする．また，$d(t_1, t_2)$ は 2 つのテキスト $t_1, t_2 \in \mathcal{T}$ 間の非類似度を評価する関数を表す．本稿を通し，テキスト t_1, t_2 をそれぞれ Sentence Transformer [21] に入力して得られる埋め込みベクトル $\boldsymbol{v}_1, \boldsymbol{v}_2$ を用いて，以下のように定義する．

$$d(t_1, t_2) = (1 - \cos(\boldsymbol{v}_1, \boldsymbol{v}_2))/2 \tag{8}$$

Algorithm 1 貪欲法に基づく例示組合せ最適化アルゴリズム

Input: 学習データセット $\mathcal{D}_{\text{train}}$, 例示集合 $\mathcal{X}_1$, $\mathcal{X}_2$, $\mathcal{X}_3$, 例示数 $\tilde{n}$, ステップ数 T

Output: 例示リスト $\hat{\mathcal{Y}}$

1: **for** $i = 1, \ldots, T$ **do**
2: $\quad \mathcal{G}^{(i)} = []$
3: $\quad$ **for** $j = 1, \ldots, \tilde{n}$ **do**
4: $\quad\quad r \sim \text{Uniform}(1, \ldots, |\mathcal{G}^{(i)}| + 1)$
5: $\quad\quad$ {$\mathcal{G}^{(i)}$ の r 番目の位置に追加する例示候補の集合 $\mathcal{X}$ を定義}
6: $\quad\quad$ **if** $r = 1$ **then**
7: $\quad\quad\quad \mathcal{X} = \mathcal{X}_1$
8: $\quad\quad$ **else**
9: $\quad\quad\quad \boldsymbol{g}_{r-1} =: [x_1, \ldots, x_m]$
10: $\quad\quad\quad$ **if** $|\boldsymbol{g}_{r-1}| = 1$ **then**
11: $\quad\quad\quad\quad \mathcal{X}' = \{x \in \mathcal{X}_2 | \phi(x) = \phi(x_1)\}$
12: $\quad\quad\quad$ **else**
13: $\quad\quad\quad\quad \mathcal{X}' = \{x \in \mathcal{X}_3 | \phi(x) = \phi(x_1)\}$
14: $\quad\quad\quad$ **end if**
15: $\quad\quad\quad \mathcal{X} = \mathcal{X}_1 \cup \mathcal{X}'$
16: $\quad\quad$ **end if**
17: $\quad\quad \hat{x} = \arg\min_{x \in \mathcal{X}} \mathcal{L}_{\text{train}}(\zeta(\alpha(\mathcal{G}^{(i)}, r, x)))$
18: $\quad\quad \mathcal{G}^{(i)} \leftarrow \alpha(\mathcal{G}^{(i)}, r, \hat{x})$
19: $\quad$ **end for**
20: **end for**
21: {T 回の例示選択結果のうち，学習損失を最小化する組合せを選択}
22: $\hat{\mathcal{G}} = \arg\min_{\mathcal{G} \in \{\mathcal{G}^{(1)}, \ldots, \mathcal{G}^{(T)}\}} \mathcal{L}_{\text{train}}(\zeta(\mathcal{G}))$
23: $\hat{\mathcal{Y}} = \zeta(\hat{\mathcal{G}})$

式 (8) の定義に基づく学習損失の最小化は，負の BERTScore [33] を用いることと等価である．式 (7) の学習損失は全学習サンプルに基づき計算されるが，本研究では計算量削減のため，アルゴリズム 1 の各 (i, j) についてランダムに選択した n_{train} 個の学習サンプルを代わりに用いて損失の計算を行うこととした．このようにして計算される学習損失を最小化する例示候補 $\hat{x}$ を集合 $\mathcal{X}$ から選択し，例示グループリスト $\mathcal{G}^{(i)}$ に追加することを繰り返すことで，$\tilde{n}$ 個の例示組合せの選択を行う．最終的に，T 回の試行で得られた例示グループリスト $\mathcal{G}^{(1)}, \ldots, \mathcal{G}^{(T)}$ から学習損失を最小化する結果 $\hat{\mathcal{G}}$ を選び，例示リスト $\hat{\mathcal{Y}}$ に変換して出力する（アルゴリズム 1 の 22–23 行目）．

上記のアルゴリズム 1 に基づき選択された例示リスト $\hat{\mathcal{Y}}$ のテスト損失を評価することで，未知のデータに対する TST のタスク性能を測ることができる．テストデータセットを

$$\begin{aligned} &\mathcal{D}_{\text{test}} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_{|\mathcal{D}_{\text{test}}|}\}, \\ &\boldsymbol{x}_i = (s_i, t_i, \tilde{s}_i, \tilde{t}_i) \text{ for } i = 1, \ldots, |\mathcal{D}_{\text{test}}| \end{aligned} \tag{9}$$

としたとき，与えられた例示リスト $\mathcal{Y}$ のテスト損失は以下で定義される．

$$\mathcal{L}_{\text{test}}(\mathcal{Y}) = \frac{1}{|\mathcal{D}_{\text{test}}|} \sum_{i=1}^{|\mathcal{D}_{\text{test}}|} d(\tilde{t}_i, f_{\text{LLM}}(\rho(\boldsymbol{x}_i, \mathcal{Y}))) \tag{10}$$

4 実　　験

本節では，パラレルデータにおける変換関係の情報の有無が TST のタスク性能に与える影響を検証することを目的として，変換関係の情報を保持する設定と保持しない設定の両ケースについて実データを用いて提案手法による例示選択を行い，それらの結果の比較を行う．

4.1 実験設定

TST の実データセットとして，Yelp データセット [11] を用いて実験を行った．本データセットに含まれるサンプルは全てパラレルデータの形式を持ち，スタイル “positive” からスタイル “negative” への変換例と，スタイル “negative” からスタイル “positive” への変換例がそれぞれ 500 個ずつ含まれる．本データセットをランダムに分割することで，3.1 節に述べたように学習データセット $\mathcal{D}_{\text{train}}$, テストデータセット $\mathcal{D}_{\text{test}}$, 例示データセット $\mathcal{D}^{(0)}_{\text{exemplar}}$ を定義した．ここで，スタイル “positive” からスタイル “negative” への変換例と，スタイル “negative” からスタイル “positive” への変換例のそれぞれについて，データセット $\mathcal{D}_{\text{train}}$, $\mathcal{D}_{\text{test}}$, $\mathcal{D}^{(0)}_{\text{exemplar}}$ のサンプルサイズをそれぞれ 225, 225, 50 とした．また，$n^{(0)} = |\mathcal{D}^{(0)}_{\text{exemplar}}|$ として，ここで得られた例示データセット $\mathcal{D}^{(0)}_{\text{exemplar}}$ を以下のように定義しておく．

$$\begin{aligned} &\mathcal{D}^{(0)}_{\text{exemplar}} = \{\boldsymbol{x}^{(0)}_1, \ldots, \boldsymbol{x}^{(0)}_{n^{(0)}}\}, \\ &\boldsymbol{x}^{(0)}_i = (s^{(0)}_i, t^{(0)}_i, \tilde{s}^{(0)}_i, \tilde{t}^{(0)}_i) \text{ for } i = 1, \ldots, n^{(0)} \end{aligned} \tag{11}$$

本研究では，例示における変換関係の情報の有無が TST の性能に与える影響を検証するため，上記の例示データセット $\mathcal{D}^{(0)}_{\text{exemplar}}$ から以下の 4 つの例示データセット $\mathcal{D}_{\text{exemplar}}$ を設定し，実験を行った（図 4）．

- **設定 ST-R**：$\mathcal{D}_{\text{exemplar}} = \mathcal{D}^{(0)}_{\text{exemplar}}$ とする．これは，元のデータセット $\mathcal{D}^{(0)}_{\text{exemplar}}$ に含まれるソース・ターゲットの両データを用いた上で，それらの変換関係の情報も考慮する（つまり，パラレルデータとして扱う）設定に相当する．
- **設定 ST**：以下の定義を用いる．これは，元のデータセット $\mathcal{D}^{(0)}_{\text{exemplar}}$ に含まれるソース・ターゲットの両データを用いるが，それらの間の変換関係の情報を考慮しない（つまり，非パラレルデータとして扱う）設定に相当する．

$$\begin{aligned} &\mathcal{D}_{\text{exemplar}} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_{2n^{(0)}}\}, \\ &\boldsymbol{x}_i = \begin{cases} (s^{(0)}_i, t^{(0)}_i) & \text{for } i = 1, \ldots, n^{(0)} \\ (\tilde{s}^{(0)}_i, \tilde{t}^{(0)}_i) & \text{for } i = n^{(0)} + 1, \ldots, 2n^{(0)} \end{cases} \end{aligned} \tag{12}$$

- **設定 S**：以下の定義を用いる．これは，元のデータセット $\mathcal{D}^{(0)}_{\text{exemplar}}$ に含まれるソーステキストのデータのみを非パラレルデータとして用いる設定に相当する．

$$\begin{aligned} &\mathcal{D}_{\text{exemplar}} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_{n^{(0)}}\}, \\ &\boldsymbol{x}_i = (s^{(0)}_i, t^{(0)}_i) \text{ for } i = 1, \ldots, n^{(0)} \end{aligned} \tag{13}$$

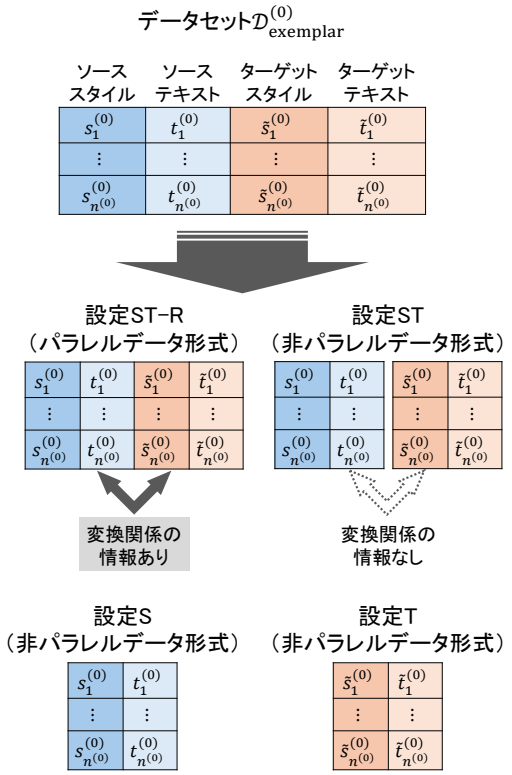

図 4　設定 ST-R, ST, S, T における例示データセットの構成.

- **設定 T**：以下の定義を用いる．これは，元のデータセット $\mathcal{D}^{(0)}_{\text{exemplar}}$ に含まれるターゲットテキストのデータのみを非パラレルデータとして用いる設定に相当する．

$$\mathcal{D}_{\text{exemplar}} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_{n^{(0)}}\},$$
$$\boldsymbol{x}_i = (\tilde{s}_i^{(0)}, \tilde{t}_i^{(0)}) \text{ for } i = 1, \ldots, n^{(0)} \tag{14}$$

上記の 4 つの設定においてそれぞれ定義された例示データセット $\mathcal{D}_{\text{exemplar}}$ から，式 (1), (2), (3) に基づき例示集合 $\mathcal{X}_1$, $\mathcal{X}_2$, $\mathcal{X}_3$ を定義した [2].

上記の例示データセット $\mathcal{D}_{\text{exemplar}}$ を用いて，アルゴリズム 1 に基づき例示組合せの最適化を行った．ここで，関数 f_{LLM} の表す言語モデルとして温度パラメータを 0 とした GPT-4o mini [7] を用い，式 (8) におけるテキストの埋め込みベクトルの計算には paraphrase-MiniLM-L6-v2 [21] を用いた．また，アルゴリズム 1 の入力における例示数を $\tilde{n} = 5$, ステップ数を $T = 5$ とし，アルゴリズム 1 の各 (i, j) において学習損失の計算に用いるサンプルサイズを $n_{\text{train}} = 20$ とした．さらに，実験では計算量削減のため，アルゴリズム 1 の各 (i, j) においてそれまでに選ばれた位置 r_{ij} の情報を軌跡 $\boldsymbol{r}^{(ij)} = [r_{i1}, \ldots, r_{ij}]$ として保持しておき，過去のステップにおいて一致する軌跡が存在する場合はその時の選択結果 $\hat{x}$ を複製して用いることとした．このようにして設定 ST-R, ST, S, T の下で選ばれた各例示組合せについて，式 (10) に基づきテスト性能の評価を行った．

2：ただし，設定 ST, S, T においては，例示データセット $\mathcal{D}_{\text{exemplar}}$ は非パラレルデータのみから構成されるため，$\mathcal{X}_2$, $\mathcal{X}_3$ は空集合とした．

4.2 実験結果

4.1 節に述べた各設定において，アルゴリズム 1 を用いて選択された例示リスト $\hat{\mathcal{Y}}$ とその学習/テスト損失をそれぞれ表 1, 2 に示す．ここで，ベースラインとして，例示なし（zero-shot）の場合の学習/テスト損失も記載した．また，表 3 に，アルゴリズム 1 における各ステップで得られた例示グループリスト $\mathcal{G}^{(1)}, \ldots, \mathcal{G}^{(T)}$ から構成される例示リスト $\zeta(\mathcal{G}^{(1)}), \ldots, \zeta(\mathcal{G}^{(T)})$ の学習/テスト損失の平均値と標準偏差を示す．

表 2, 3 より，ソース・ターゲットの両データを用いた上で，それらの変換関係の情報も考慮する設定 ST-R が最良のテスト性能を達成することが示された．また，同じくソース・ターゲットの両データを用いた場合でも，それらの変換関係の情報を考慮しない設定 ST においては，ソーステキストのみ/ターゲットテキストのみのデータを用いた場合（設定 S/T）と同程度のテスト性能しか達成されなかった．これらの結果から，本稿における実験設定の下では，高精度な TST を実現するためにパラレルデータ形式の例示が必要であることが示された．

また，表 1 から，実際に各設定の下で選択された例示組合せを比較することができる．例えば，設定 ST-R の下で選択された例示組合せにおいては，3, 4 番目にパラレルデータ形式の例示の組が選択されていることが分かる．また，ソーステキストのデータを含む設定 ST-R, ST, S のすべての場合において，例示として共通のテキスト “Here is a text, which is negative: {it did n’t get finished .}.” が選択されている．この他にも，テキスト “Here is a text, which is negative: {worst chicken parmesan i have ever had.}.,” “Here is a text, which is negative: {not one of my regular spots in scottsdale}.” は複数の設定下で例示として選択されている．これらの結果から，例示データセットの中でも学習損失の最小化に寄与するような特定の例示の存在が示唆された．

5 おわりに

本研究では，事前学習済み LLM を用いて ICL の枠組みに基づき TST タスクを解くアプローチにおいて，例示組合せの最適化を行うアルゴリズムを提案した．特に，TST タスクにおいてはパラレル・非パラレルの 2 種類の形式に基づくデータが存在することに着目し，これらの形式の違いがタスク性能に与える影響を検証するため，両形式のデータセットに関して得られる例示選択の結果を統一的に評価可能な枠組みを提案した．本研究における実験設定の下では，高精度な TST を実現するためにパラレルデータにおけるテキストの変換関係の情報が必要であることが示された．

本研究においては，例示数 $\tilde{n}$ を固定した上で式 (10) の評価規準に基づき異なる設定下でのタスク性能を比較したが，異なる例示数の設定や，BLEU [18] などの異なる評価規準を用いた場合についても同様の検証を行うことは，今後の課題である．また，TST タスクにおいて最良の性能を達成可能な例示数を選択することも，重要な課題である．

表 1　各設定において選択された例示組合せの結果.

設定	選択された例示リスト
ST-R	Here is a text, which is negative: {safeway has officially lost my business to sprouts , & fresh & easy .}.
	Here is a text, which is negative: {it did n't get finished .}.
	Here is a text, which is positive: {my husband and i enjoyed our 3rd anniversary here .}.
	Here is a rewrite of the text, which is negative: {my husband and i didn't enjoy our 3rd anniversary hear .}.
	Here is a text, which is negative: {as soon as they delivered i was like ugh .}.
ST	Here is a text, which is negative: {it did n't get finished .}.
	Here is a text, which is negative: {worst chicken parmesan i have ever had.}.
	Here is a text, which is negative: {not one of my regular spots in scottsdale}.
	Here is a text, which is negative: {fish tacos were the worst I had}.
	Here is a text, which is positive: {Ra was a chain, wow im impressed}.
S	Here is a text, which is negative: {it did n't get finished .}.
	Here is a text, which is negative: {i 'm sure they must get it right some days but not this day .}.
	Here is a text, which is positive: {i loved the ribs more than the chicken .}.
	Here is a text, which is negative: {in turn my legs are burnt from the noodles and all over the floor .}.
	Here is a text, which is positive: {the lunch and dinner items are very good as well .}.
T	Here is a text, which is positive: {I'm one of the corn people. }.
	Here is a text, which is negative: {not one of my regular spots in scottsdale}.
	Here is a text, which is negative: {worst chicken parmesan i have ever had.}.
	Here is a text, which is negative: {came here without my family .}.
	Here is a text, which is positive: {The short rib hash was perfectly cooked and juicy.}.

表 2　選択された例示リスト $\hat{\mathcal{Y}}$ の学習/テストデータにおける損失.

設定	学習損失	テスト損失
ST-R	**0.1635**	**0.1680**
ST	0.1840	0.1979
S	0.1894	0.1945
T	0.1920	0.2058
Zero-shot	0.2012	0.2050

表 3　T 回の例示選択結果の学習/テストデータにおける損失 ($T = 5$).

設定	学習損失	テスト損失
ST-R	$\mathbf{0.1722 \pm 0.0094}$	$\mathbf{0.1685 \pm 0.0004}$
ST	0.1888 ± 0.0032	0.2001 ± 0.0014
S	0.1928 ± 0.0038	0.1930 ± 0.0009
T	0.1963 ± 0.0034	0.2051 ± 0.0006

文　献

[1] Magdalena Badura, Michał Lampert, and Rafał Dreżewski. System supporting poetry generation using text generation and style transfer methods. *Procedia Computer Science*, Vol. 207, pp. 3310–3319, 2022.

[2] Jiale Cheng, Xiao Liu, Kehan Zheng, Pei Ke, Hongning Wang, Yuxiao Dong, Jie Tang, and Minlie Huang. Black-box prompt optimization: Aligning large language models without model training. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 3201–3219, 2024.

[3] Mingkai Deng, Jianyu Wang, Cheng-Ping Hsieh, Yihan Wang, Han Guo, Tianmin Shu, Meng Song, Eric Xing, and Zhiting Hu. RLPrompt: Optimizing discrete text prompts with reinforcement learning. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pp. 3369–3391, 2022.

[4] Qingxiu Dong, Lei Li, Damai Dai, Ce Zheng, Jingyuan Ma, Rui Li, Heming Xia, Jingjing Xu, Zhiyong Wu, Baobao Chang, Xu Sun, Lei Li, and Zhifang Sui. A survey on in-context learning. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 1107–1128, 2024.

[5] Zhenxin Fu, Xiaoye Tan, Nanyun Peng, Dongyan Zhao, and Rui Yan. Style transfer in text: exploration and evaluation. *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 32, No. 1, 2018.

[6] Zhiting Hu, Zichao Yang, Xiaodan Liang, Ruslan Salakhutdinov, and Eric P. Xing. Toward controlled generation of text. In *Proceedings of the 34th International Conference on Machine Learning*, Vol. 70, pp. 1587–1596, 2017.

[7] Aaron Hurst, et al. GPT-4o system card. arXiv:2410.21276, 2024.

[8] Yasaman Jafari, Dheeraj Mekala, Rose Yu, and Taylor Berg-Kirkpatrick. MORL-prompt: An empirical analysis of multi-objective reinforcement learning for discrete prompt optimization. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pp. 9878–9889, 2024.

[9] Harsh Jhamtani, Varun Gangal, Eduard Hovy, and Eric Nyberg. Shakespearizing modern language using copy-enriched sequence to sequence models. In *Proceedings of the Workshop on Stylistic Variation*, pp. 10–19, 2017.

[10] Vineet John, Lili Mou, Hareesh Bahuleyan, and Olga Vechtomova. Disentangled representation learning for non-parallel text style transfer. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 424–434, 2019.

[11] Juncen Li, Robin Jia, He He, and Percy Liang. Delete, retrieve, generate: a simple approach to sentiment and style transfer. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pp. 1865–1874, 2018.

[12] Fuli Luo, Peng Li, Pengcheng Yang, Jie Zhou, Yutong Tan, Baobao Chang, Zhifang Sui, and Xu Sun. Towards fine-

grained text sentiment transfer. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 2013–2022, 2019.

[13] Chenyang Lyu, Zefeng Du, Jitao Xu, Yitao Duan, Minghao Wu, Teresa Lynn, Alham Fikri Aji, Derek F. Wong, and Longyue Wang. A paradigm shift: The future of machine translation lies with large language models. In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, pp. 1339–1352, 2024.

[14] Do June Min, Veronica Perez-Rosas, Ken Resnicow, and Rada Mihalcea. VERVE: Template-based ReflectiVE rewriting for MotiVational IntErviewing. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pp. 10289–10302, 2023.

[15] Shervin Minaee, Tomas Mikolov, Narjes Nikzad, Meysam Chenaghlu, Richard Socher, Xavier Amatriain, and Jianfeng Gao. Large language models: A survey. arXiv:2402.06196, 2024.

[16] Sourabrata Mukherjee, Vojtěch Hudeček, and Ondřej Dušek. Polite chatbot: A text style transfer application. In *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics: Student Research Workshop*, pp. 87–93, 2023.

[17] Sourabrata Mukherjee, Atul Kr. Ojha, and Ondrej Dusek. Are large language models actually good at text style transfer? In *Proceedings of the 17th International Natural Language Generation Conference*, pp. 523–539, 2024.

[18] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. BLEU: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pp. 311–318, 2002.

[19] Reid Pryzant, Dan Iter, Jerry Li, Yin Lee, Chenguang Zhu, and Michael Zeng. Automatic prompt optimization with "gradient descent" and beam search. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 7957–7968, 2023.

[20] Emily Reif, Daphne Ippolito, Ann Yuan, Andy Coenen, Chris Callison-Burch, and Jason Wei. A recipe for arbitrary text style transfer with large language models. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pp. 837–848, 2022.

[21] Nils Reimers and Iryna Gurevych. Sentence-BERT: Sentence embeddings using Siamese BERT-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pp. 3982–3992, 2019.

[22] Pasi Shailendra, Rudra Chandra Ghosh, Rajdeep Kumar, and Nitin Sharma. Survey of large language models for answering questions across various fields. In *2024 10th International Conference on Advanced Computing and Communication Systems (ICACCS)*, Vol. 1, pp. 520–527, 2024.

[23] Tianxiao Shen, Tao Lei, Regina Barzilay, and Tommi Jaakkola. Style transfer from non-parallel text by cross-alignment. In *Advances in Neural Information Processing Systems*, Vol. 30, pp. 6833–6844, 2017.

[24] Mirac Suzgun, Luke Melas-Kyriazi, and Dan Jurafsky. Prompt-and-rerank: A method for zero-shot and few-shot arbitrary textual style transfer with small language models. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pp. 2195–2222, 2022.

[25] Jonathan Tonglet, Manon Reusens, Philipp Borchert, and Bart Baesens. SEER : A knapsack approach to exemplar selection for in-context HybridQA. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 13569–13583, 2023.

[26] Martina Toshevska and Sonja Gievska. A review of text style transfer using deep learning. *IEEE Transactions on Artificial Intelligence*, Vol. 3, No. 5, pp. 669–684, 2022.

[27] Martina Toshevska and Sonja Gievska. LLM-based text style transfer: Have we taken a step forward? *IEEE Access*, Vol. 13, pp. 44707–44721, 2025.

[28] Xingchen Wan, Ruoxi Sun, Hootan Nakhost, and Sercan Ö. Arık. Teach better or show smarter? on instructions and exemplars in automatic prompt optimization. In *Advances in Neural Information Processing Systems*, Vol. 37, pp. 58174–58244, 2024.

[29] Jianyu Wang, Zhiqiang Hu, and Lidong Bing. Evolving prompts in-context: An open-ended, self-replicating perspective. In *Forty-second International Conference on Machine Learning*, 2025.

[30] Zhaoxuan Wu, Xiaoqiang Lin, Zhongxiang Dai, Wenyang Hu, Yao Shu, See-Kiong Ng, Patrick Jaillet, and Bryan Kian Hsiang Low. Prompt optimization with EASE? efficient ordering-aware automated selection of exemplars. In *Advances in Neural Information Processing Systems*, Vol. 37, pp. 122706–122740, 2024.

[31] Qinyuan Ye, Mohamed Ahmed, Reid Pryzant, and Fereshte Khani. Prompt engineering a prompt engineer. In *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 355–385, 2024.

[32] Haopeng Zhang, Philip S. Yu, and Jiawei Zhang. A systematic survey of text summarization: From statistical methods to large language models. *ACM Computing Surveys*, Vol. 57, No. 11, 2025.

[33] Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q. Weinberger, and Yoav Artzi. BERTScore: Evaluating text generation with BERT. In *International Conference on Learning Representations*, 2020.

[34] Yongchao Zhou, Andrei Ioan Muresanu, Ziwen Han, Keiran Paster, Silviu Pitis, Harris Chan, and Jimmy Ba. Large language models are human-level prompt engineers. In *The Eleventh International Conference on Learning Representations*, 2023.