Vol.2022-NL-253 No.12
2022/9/30

スタイル分離に基づくスタイル変換と異常検知の同時学習に基づく文書のスタイル一貫性の改善

京野 長彦[1,a)]　吉永 直樹[2,b)]　佐藤 翔悦 * [2,c)]

概要：文書のスタイルは文書全体で一貫していることが望ましいが，意図せず不適切なスタイルの文が混入することも多い．そこで本研究では，文のスタイル分離に基づくスタイル変換器と自己教師あり学習に基づく異常検知器の同時学習を用いて，文書のスタイル一貫性を改善する手法を提案する．具体的にはまず既存のスタイル変換データセットを用い，対応関係にあるスタイルの文を無作為に組み合わせて擬似的にスタイルの一貫しない文書（文集合）を自動構築する．このようにして自動構築した文書を学習データとして用いて，その入力文書に含まれる各文のスタイルをベクトル表現として分離し，教師なしスタイル変換器と Transformer に基づく教師あり異常検知器にそれぞれ入力して各モデルを同時学習させる．推論時には後者の異常検知器を用いて異質なスタイルで書かれた文を検出し，検出された文のスタイルを，前者のスタイル変換器によって，入力中の他の文のスタイルを考慮しながら変換する．実験として，4 種類のスタイル変換データセットを用いて，上記の方法で人工的な学習・評価データを構築して性能を評価し，さらに実際の文書から構築した評価データセットを用いて実践的な評価を行う．

1. はじめに

一般的に，人が読む文書は，読み手や内容に適したスタイルで一貫して書かれていることが望ましい．例えば，目上の人へのメールで無礼な表現を使ってしまったり，相手の解しない方言や，特定のコミュニティで使われる用語を混ぜてしまったりすると，諍いや誤解が生まれかねない．

一方で，機械学習モデルで自然言語処理タスクを解く際も，学習データ，あるいは評価データと学習データのスタイルの不一致が問題を引き起こすことがある．具体的に，機械翻訳や対話システムなど言語生成タスクでは，学習・評価データに異質なスタイルのデータが混入することで，性能の劣化やスタイルの一貫しない出力が得られることがある [1], [2]．さらにはスタイル変換を用いて学習データや評価データを改変することで，モデルが正しく学習・推論できなくする攻撃手法も発見されている [3]．

このような背景を受けて，与えられた文を指定されたスタイルに変換するスタイル変換タスクが広く取り組まれている [4], [5], [6]．しかしながら，既存のスタイル変換の研究では，基本的に変換単位は文で，入出力のスタイルは既知であることを仮定している [7]．そのため，既存のスタイル変換技術を文書の一貫性の改善に用いるには，どの文のスタイルを，どのように書き換えるかを指定する必要がある．これは，方言や専門用語など，書き換えるべき表現にユーザ自身が気がつかないことが多いことを考えると現実的ではない．また，文書のスタイルはドメインや個々のユーザに依存して多岐にわたるため，言語資源の存在する典型的なスタイルにしか対応できない既存のスタイル変換では，入力文書ごとに異なる出力スタイルに柔軟に対応することが難しい．

そこで本研究では入力文書のスタイル一貫性を改善するタスク（**図 1**）[8] に取り組み，具体的に Transformer [9] に基づく教師あり異常検知モデルでスタイルが一貫しない文を検知した上で，その文以外の均質なスタイルの文を目

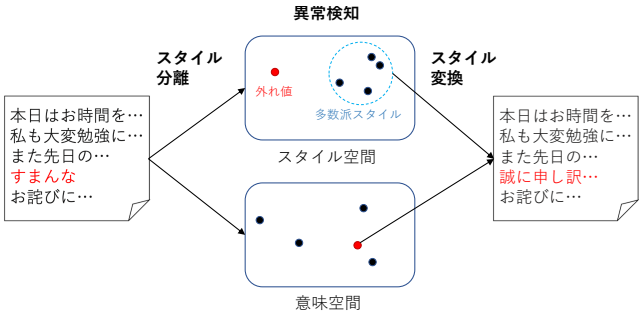


図 1: 異常検知に基づく文書スタイルの一貫性改善．

*現在は企業に所属．
1 東京大学大学院 情報理工学系研究科
2 東京大学 生産技術研究所
a) kyono@tkl.iis.u-tokyo.ac.jp
b) ynaga@iis.u-tokyo.ac.jp
c) shoetsu@tkl.iis.u-tokyo.ac.jp

的スタイルとして参照して，検知した文のスタイルを教師なしで変換する手法を提案する．先行研究 [8] では教師なし異常検知手法を用いて異質なスタイルの文を検出していたが，異常検知をするのに適したスタイルベクトルが必ずしも計算できておらず，十分な検出性能を発揮できていなかった．そこで教師あり異常検知器を用いてスタイル一貫性に関わる特徴量を学習することで，異常なスタイルの発見性能を改善し，スタイル一貫性の改善タスクの改良を試みる．さらに，提案モデルを多様なスタイルに対応させることを狙って，複数の既存のスタイル変換データセットを用いてモデルを訓練する．

実験では，既存のスタイル変換データセットである Sarcasm [10]，GYAFC [11]，ShortJoke[*1]，Hate Speech and Offensive Language データセット [12] のそれぞれから無作為に文を組み合わせて擬似文書の学習・開発・評価データを自動構築し，提案手法のスタイル判定・変換性能に関する分析と考察を行う．加えて実際の文書を収集した Amazon Review Data [13] を元に評価データを構築し，同様に分析と考察を行う．

本論文の貢献を以下に示す．

- 教師あり異常検知モデルとスタイル変換モデルの同時学習により文書スタイルの一貫性を改善する手法を提案
- 文書スタイルの一貫性改善タスクについて，文書中の多数派スタイルの割合に基づく評価尺度を提案
- 文書スタイルの一貫性を改善する手法を評価するためのデータセットを実文書から構築
- スタイル一貫性改善タスクについて，擬似文書・実文書に基づくデータセットを用いて，手法の有効性を評価

2. 関連研究

本節では，既存のスタイル変換手法について，タスク設定とスタイル一貫性の改善タスクへの適用可能性を中心に議論する．続いて，我々が先行研究で用いたスタイル分離と教師なし異常検知に基づくスタイル変換手法について説明する．

2.1 テキストのスタイル変換

既存のテキストのスタイル変換手法では，基本的に文を入出力の単位とするスタイル変換に取り組んでおり，既定の入出力スタイルのペアに対応する学習データからそのスタイル専用のモデルを学習するものが多い [14], [15], [16]．このシステムを文書のスタイル一貫性の改善に用いる場合，2 つの問題がある．1 つは実際の文書では適切なスタイルと変換すべきスタイルの文が混在し，しかも変換すべきスタイルが文書中のどの文であるか与えられず，しかもユーザもそれがわかるとは限らないことである．もう 1 つは入出力として未知のスタイルを処理する必要性が生じた場合，適切なスタイル表現を計算できないことである．

本研究と同様に，複数文の入力を想定したスタイル変換手法としては，Cheng らの研究が存在する [7]．しかしこの手法は，変換対象の周辺文脈を考慮することで，変換対象とする文に対するスタイル変換性能の改善を目指したものであり，変換対象の文が予め指定されていなければならない．それゆえ，前述の 2 つの問題に対処することができない．

2.2 文書のスタイル一貫性の改善手法

我々は，2.1 節で述べた問題点を受けて，先行研究 [8] で文書中の文の過半数が均質なスタイルで書かれた文（集合）（以後，“多数派”）に少数の異質なスタイルの文（以後，“少数派”）が混入した文書を入力し，入力文書中の少数派のスタイルを多数派のものへと変換するタスクを提案した．さらに，そのタスクを実現するための手法として，入力中の各文から教師なし異常検知手法を用いて少数派を検出し，教師なしスタイル変換手法を用いて多数派のスタイルに変換する枠組みを提案した．この枠組みに基づいてスタイル変換を行うことで，前述の二つの問題に対処することが可能である．しかしながら，人工データセットを用いた予備実験において，先行研究の手法では，少数派のスタイルの文を適切に検出することができず，スタイルの一貫性を改善することもできなかった．先行研究では，少数派スタイルの文の検出に密度に基づく教師なし異常検知手法 [17] を用いていたが，この異常検知手法は高次元ベクトルを入力した場合の動作が不安定であることが知られている [18]．また，スタイル変換・分離器自体も異常検知を前提として最適化されていない．以上の理由から，先行研究では十分な検出性能が達成できなかったと考えられる．

そこで，本研究では，先行研究の枠組みにおいて，異常検知に Transformer [9] をベースとした教師あり異常検知器を用い，スタイル分離・変換手法と同時学習することで変換性能を改善することを目指す．また，実文書を用いた評価を行い，提案手法の実際的な有効性を検証する．

3. 提案手法

本研究で提案する文書スタイルの一貫性改善手法は，先行研究 [8] と同様に少数派スタイルを検出するモジュールと，スタイル変換を行うモジュールの 2 つから構成されており，前者で教師あり学習に基づくモデルを採用したところに違いがある．これらのモジュールは，文のスタイル表現ベクトルを計算するスタイルエンコーダを共有している．この 2 モジュールを多様なスタイル変換データセットを元に構築した人工データを用いて同時に学習することで，ス

[*1] https://github.com/amoudgl/short-jokes-dataset

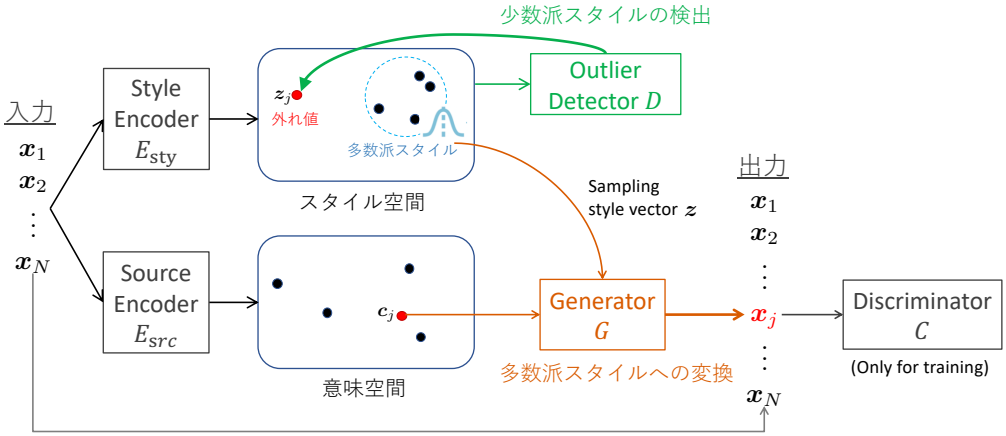


図 2: 提案モデルの全体図．学習データ中の各文にはスタイルタグが付与されていることを前提とし，訓練時には異常検知器の出力を用いずタグから少数派の検知を行い，スタイル変換を行う．推論時にはそのタグを用いず，異常検知器による判定結果から少数派スタイルを検知し，スタイル変換を行う．なお，少数派スタイルの文が複数検出された場合は，各文ごとにスタイルを変換する．

タイル変換とスタイル異常検知の両タスクに対してバランスよく最適化されたスタイル表現を獲得し，検出精度と変換性能を同時に改善することを目指す．提案手法の全体像を**図 2** に示す．

以下では，タスクの入力文書を $X = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_N\}$ と表記する．ただし，$\boldsymbol{x}_j \in X$ は入力中の各文（トークン列），N は入力文書中の文数である．また，多数派スタイルを s，少数派スタイルを $\bar{s}$ とする．学習時にはスタイルの逆変換を行うため，それに必要な少数派スタイル $\bar{s}$ を持つ文のみから成る文集合を別途用意し，$\bar{X}$ と表記する．以下，両モジュールについて説明する．

3.1 異常検知モジュール

異常検知モジュールはスタイルエンコーダ E_{sty} と異常検知器 D から構成される（図 2）．スタイルエンコーダ E_{sty} は入力として 1 文を取り，対応するスタイルベクトルを出力する．入力文書 X が与えられるとき，X の各文ををスタイルエンコーダに入力して得られたスタイルベクトルの集合を $Z = \{\boldsymbol{z}_1, \boldsymbol{z}_2, \cdots, \boldsymbol{z}_N\}$ とする．

異常検知器 D は Transformer [9] エンコーダと 2 層の多層パーセプトロン（MLP）から構成されている．このモジュールはスタイルエンコーダによって得られた各文に対応するスタイル表現ベクトル Z を入力とし，それぞれのベクトルに対してそれが外れ値である確率 $P = \{p_1, p_2, \cdots, p_N\}$ を出力する，各文単位で 2 値分類の系列ラベリングを行うモデルである．損失には，式 1 で表される binary cross entropy loss を用いる．

$$\mathcal{L}_{detec} = -\sum_{j=1}^{N} \{p_j \cdot \log l_j + (1 - p_j) \cdot \log(1 - l_j)\} \quad (1)$$

ただし，l_j は $\boldsymbol{x}_j$ が少数派であるかどうかを示すラベルである（$l_j = 1$ のとき外れ値とする）．

3.2 スタイル変換モジュール

本研究では先行研究 [8] と同様に，スタイル変換モデルとして，敵対的学習に基づくスタイル変換モデル StyIns [19] を採用した．StyIns は変換先のスタイルを持つ文集合をスタイル参照文として入力にとるため，文書中の多数派の文をスタイル参照文とすることで，変換先スタイルを動的にモデルに指示することができると期待される．提案手法の実装にあたっては，Yi らによる実装[*2]を改変して用いる．

StyIns は図 2 の各モジュールのうち，スタイルエンコーダ E_{sty}，ソースエンコーダ E_{src}，生成器 G，識別器 C から成る．スタイルエンコーダは 3.1 節で説明したスタイルエンコーダと同一のものである．スタイル変換モジュールは入力として変換対象を 1 文と，変換先のスタイルを持つスタイル参照文を複数文とり，変換対象の文をスタイル参照文と同じスタイルに変換したものを出力する．ここでは入力文書 X から検出された少数派を $X^{minor} = \{\boldsymbol{x}_j | \boldsymbol{x}_j \in X, l_j = 1\}$ とし，その数を $N^{minor} = |X^{minor}|$ と表記する．X のうちスタイル参照文となる多数派を $X^{major} = X \setminus X^{minor}$，その分数を $N^{major} = N - N^{minor}$ と表記する．変換対象となる 1 文を $\boldsymbol{x}_j^{minor} \in X^{minor}$ とする．

(1) 変換対象 $\boldsymbol{x}_j^{minor}$ をソースエンコーダ E_{src} に入力し，文の意味ベクトル $\boldsymbol{c}_j$ を計算する．
(2) スタイル参照文集合 X^{major} をスタイルエンコーダ E_{sty} に入力し，各文に対応するスタイル表現ベクトル $Z^{major} = \{\boldsymbol{z}_1, \boldsymbol{z}_2, \cdots, \boldsymbol{z}_{N^{major}}\}$ を得る．
(3) Z^{major} の平均と分散を計算し，その値を平均・分散とする正規分布からスタイル表現 $\boldsymbol{z}$ をサンプルする．
(4) $\boldsymbol{c}_j$ および分布からサンプルしたスタイル表現 $\boldsymbol{z}$ を生成器 G に入力し，変換後の文 $\boldsymbol{y}_j$ を得る．
(5) $\boldsymbol{x}_j \in X^{major}$ については，$\boldsymbol{y}_j = \boldsymbol{x}_j$ とする．

[*2] https://github.com/XiaoyuanYi/StyIns

表 1: データセット毎に作成した学習・開発・評価データの情報

(a) スタイル変換データセットが含むスタイルと統計量.

データセット	スタイル	文数	平均トークン数
Sarcasm	Sarcastic/Normal	402,000	12.7
GYAFC	Formal/Informal	224,000	12.5
ShortJoke	Jocular/Normal	463,000	16.6
HSOL	Offensive/Non-Offensive	25,000	20.6

(b) スタイル変換データセットから構築した学習・開発・評価データの統計量.

データセット	学習			開発			評価		
	文書数	平均文数	平均トークン数	文書数	平均文数	平均トークン数	文書数	平均文数	平均トークン数
Sarcasm	80,000	10	122.3	2,000	10	124.3	1,000	10	122.4
GYAFC	80,000	10	125.7	2,000	10	121.6	1,000	10	125.5
ShortJoke	80,000	10	166.5	2,000	10	165.4	1,000	10	167.6
HSOL	-	-	-	-	-	-	1,000	10	229.1

損失には，入力 $\boldsymbol{x}_j^{minor}$ を $\boldsymbol{x}_j^{minor}$ 自身に変換した際の一致度から計算する式 2 の reconstruction loss，出力 $\boldsymbol{y}_j$ に対し逆変換を行い $\boldsymbol{x}_j^{minor}$ との一致度から計算する式 3 の cycle consistency loss，出力 $\boldsymbol{y}_j$ と分類器 C からスタイルに関する敵対的学習を行う式 4 の adversarial style loss の 3 つを用いる．

$$\mathcal{L}_{recon} = -\log p_G(\boldsymbol{x}_j^{minor}|\boldsymbol{x}_j^{minor}, \bar{X}) \quad (2)$$

$$\mathcal{L}_{cycle} = -\log p_G(\boldsymbol{x}_j^{minor}|\boldsymbol{y}_j, \bar{X}) \quad (3)$$

$$\mathcal{L}_{style} = -\log p_C(s|\boldsymbol{y}_j) \quad (4)$$

3.3 異常検知とスタイル変換の同時学習

異常検知に適したスタイル表現を得るために，式 1 から式 4 の和である $\mathcal{L}_{detec} + \mathcal{L}_{recon} + \mathcal{L}_{cycle} + \mathcal{L}_{style}$ を最小化して 2 つのモジュールを同時に学習する．同時学習を行わない場合には，まず $\mathcal{L}_{recon} + \mathcal{L}_{cycle} + \mathcal{L}_{style}$ を最小化してスタイル変換モジュールの学習を行い，その後，スタイルエンコーダのパラメタを固定して，$\mathcal{L}_{detec}$ を最小化し，異常検知モジュールを学習する．

4. 実験設定

提案手法の有効性を評価するために，人工データと実データを用いて異常検知性能とスタイル変換性能の評価を行う．提案手法の学習には既存のスタイル変換データセットから構築した人工データを用いるが，これらのデータに含まれない学習外スタイルに対する評価も合わせて行う．

4.1 データ

本節では，4.1.1 節で提案手法の学習・開発に用いる人工データ，および，評価に用いる人工データを生成する手法を説明する．次に，4.1.2 節で評価に用いる実データを半自動収集する手順について説明する．

4.1.1 学習・開発・評価用人工データ

本研究では，先行研究 [8] と同様に，formal と informal のような対になる 2 スタイルの文集合から構成される既存のスタイル変換データセットを利用し，人工的に異なるスタイルが混在した擬似文書データを手法の学習・開発・評価データとして生成する．実際的な文書での評価を想定し，各データセットが含む 2 つのスタイルから，一方を多数派，もう一方を少数派とみなして文をランダムに組み合わせ，少数派を含む擬似文書を作成する．最終的に得られた文書集合をランダムに 80:2:1 に分割して，提案モデルの学習・開発データと，学習内スタイルに対する人工の評価データを得る．

擬似文書集合はモデルの学習にも利用するため，学習外スタイルに適用することを念頭に，多様なスタイル変換データセットを元に，出力スタイルの分布に偏りが生じないよう留意して構築した．具体的に，スタイル一貫性改善タスクでは，意味を改変するスタイル変換は想定しないことを考慮して，意味を保持するスタイル変換データセットとして Sarcasm [10]，GYAFC [11]，ShortJoke[*1]を利用して擬似文書集合を構築した．各データセットは**表 1a** のように，それぞれ対となる 2 つのスタイルの文集合から構成されている．この中で GYAFC と ShortJoke は，Kang らによる xSLUE ベンチマーク [20] 用の前処理器によって前処理が施されたものを用いる．

スタイル変換データセットから擬似文書を構築する具体的な手順は以下の通りである．まず，各スタイルごとに 80,000 文，2,000 文，1,000 文を学習・開発・評価文書作成のためのシード文としてサンプルする．以下，データセッ

ト内の対になっているスタイルのうち，多数派とする方のスタイルをスタイル A，少数派とする方のスタイルをスタイル B と呼称する．スタイル A と B の文をそれぞれ l 文サンプルし，それぞれ X_A，X_B と呼称する[*3]．異常検知の学習時には，X_A と X_B の文を m 文ランダムに入れ替える[*4]ことで，複数の少数派が混ざった擬似文書を構築する．スタイル変換の学習時には，X_B からランダムに 1 文 $\boldsymbol{x}_B$ を選んで変換対象とし，X_A から 1 文ランダムに除いて残りをスタイル参照文とする．また X_B のうち選ばれなかった文集合 $X_B \setminus \boldsymbol{x}_B$ を式 2 や式 3 での逆変換で用いるスタイル参照文とする．得られた文書 N 件をランダムに 80:2:1 に分割して，学習・開発・評価データを得る．得られたデータの統計量を**表 1b** に示す．

学習外スタイルで書かれた文書に対する異常検知およびスタイル変換の性能を確認するため，Hate Speech and Offensive Language（以後，HSOL）[12] を xSLUE で前処理したものを用いて 4.1.1 節と同様の処理を行った．得られたデータの詳細を表 1a および表 1b に示す．

4.1.2 実文書から収集した評価データ

4.1.1 節で構築した評価データに加えて，Amazon.com のレビュー [13] から実際的な評価データを収集する．具体的には，Amazon Review Data[*5]の Video Games small subsets を用いて，学習内スタイルである GYAFC，学習外スタイルである HSOL それぞれに対応する評価データセットを 1 つずつ計 2 種類作成する．以下では，それぞれ AmazonFormality，AmazonOffensiveness と呼称する．Amazon Review Data にはどの文がスタイルの異なる文であるかのラベルは付与されていないため，GYAFC から学習したスタイル判定器および HSOL から学習したスタイル判定器を適用して，2 つのスタイルを含むレビューを抽出したのち，そのレビューを対象として人手でラベルの修正を行い，評価用データを構築した．

スタイル判定器は，事前学習済み BERT モデル [21] である `bert-base-uncased`[*6]を GYAFC に対し fine-tuning したものと，HSOL に対し fine-tuning したものを用いる．BERT 判定器のモデルとしては Transformers [22] (v4.20.1) の `BertForSequenceClassification` クラスを使用する．事前実験におけるこの判定器の分類精度は GYAFC で 89.5%，HSOL で 97.2%であり，スタイル変換の評価用擬似ラベルとしては十分な精度であると考えている．最終的に得られた AmazonFormality，AmazonOffensiveness の統計量を**表 2** に示す．

[*3] $l \sim \mathcal{N}(10, 4)$．ただし，$l$ は 3 を下限とする．
[*4] $m \sim \mathcal{N}(1, 1)$．ただし，$m < \frac{l}{2}$ とする．
[*5] https://nijianmo.github.io/amazon/index.html
[*6] https://huggingface.co/bert-base-uncased

表 2: 実文書から収集した評価データの統計量．平均文数は 1 文書内の文数の平均．

データセット	評価		
	文書数	平均文数	平均トークン数
AmazonFormality	200	8.1	145.4
AmazonOffensiveness	200	9.2	158.2

4.2 実装およびハイパーパラメタ

本研究では，スタイル変換モデル StyIns [19] を基本として異常検知モジュールを追加し，提案モデルを実装する．各手法はすべて PyTorch (v1.11.0) [23] 上に実装を行った．StyIns は Yi らの実装[*2]をベースに変更を行った．本来，StyIns のスタイルエンコーダは 2 文以上の入力を前提として分布を出力する仕様となっているが，本研究では 1 文に対して単一のスタイルベクトルを計算する必要があるため（3.1 節），このエンコーダを拡張し，入力が 1 文のみの際に分布の平均をそのままスタイル表現ベクトルとして出力する実装とした．

StyIns のハイパーパラメタは Yi らの実装に倣ったが，バッチサイズを 128 とし，語彙数を 50,000 に制限した．異常検知モジュールの Transformer エンコーダや optimizer の設定は，Vaswani らによる設定 [9] を模倣した．ただし，StyIns のスタイルエンコーダの次元数が 1,024 次元であるため，Transformer エンコーダのヘッド数はそれを割り切れる 8 に変更し，過学習を防ぐため，学習率を 0.5 倍し，式 5 の通りにした．

$$lerning\_rate = 0.5 d_{\text{model}}^{-0.5} \cdot \min(step\_num^{-0.5}, step\_num \cdot warmup\_steps^{-1.5}) \quad (5)$$

ただし，d_{model} は Transformer への入力の次元数であり，本研究では $d_{\text{model}} = 1,024$ である．異常検知モジュールとスタイル変換モジュールはそれぞれ 20 エポックの訓練を行った．

4.3 比較モデル

実験では提案モデル以外に，ベースラインとして以下の手法を学習用人工データで訓練したモデルを比較する．

StyIns + LOF [8]: 異常検知モジュールとして教師なし異常検知手法である Local Outlier Factor (LOF) [17] を用いる．LOF は scikit-learn (v1.1.1) [24] の実装を用いる．LOF の nearest neighbors の数は入力文書の文数の半分，contamination の割合は自動判定とする．

StyIns + Transformer ：提案モデルに含まれる異常検知モジュールとスタイル変換モジュールの同時学習を行わず，スタイル変換のみの学習をした後にスタイルエンコーダのパラメタを固定して，Outlier Detector のみの学習を行う．

表 3: 少数派スタイル文の検出性能（擬似文書）．小字は標準偏差．

少数派スタイル検知	Sarcasm			GYAFC			ShortJoke		
	P	R	F_1	P	R	F_1	P	R	F_1
LOF	$0.316_{0.023}$	$0.723_{0.030}$	$0.439_{0.023}$	$0.289_{0.019}$	$0.560_{0.065}$	$0.379_{0.013}$	$0.264_{0.043}$	$0.349_{0.041}$	$0.297_{0.023}$
Transformer	$\mathbf{0.772}_{0.035}$	$0.649_{0.034}$	$0.704_{0.015}$	$\mathbf{0.769}_{0.036}$	$0.443_{0.023}$	$0.562_{0.019}$	$\mathbf{0.799}_{0.035}$	$0.584_{0.045}$	$0.674_{0.031}$
Transformer（同時学習）	$0.708_{0.016}$	$\mathbf{0.872}_{0.009}$	$\mathbf{0.782}_{0.010}$	$0.721_{0.028}$	$\mathbf{0.624}_{0.027}$	$\mathbf{0.668}_{0.013}$	$0.736_{0.018}$	$\mathbf{0.895}_{0.019}$	$\mathbf{0.808}_{0.017}$

少数派スタイル検知	HSOL		
	P	R	F_1
LOF	$\mathbf{0.160}_{0.028}$	$0.124_{0.056}$	$0.136_{0.037}$
Transformer	$0.140_{0.016}$	$0.084_{0.036}$	$0.103_{0.032}$
Transformer（同時学習）	$0.124_{0.007}$	$\mathbf{0.248}_{0.028}$	$\mathbf{0.165}_{0.010}$

またスタイル変換の性能評価時のベースラインとして，異常検知を行わない手法（+検知なし），および正解の外れ値を与える手法（+Oracle）とも比較する．

無変換 + 検知なし ：入力を変換せずそのまま出力する．

StyIns + 検知なし ：全文をスタイルエンコーダに入力してスタイル表現を計算し，全文を変換する．

StyIns + Oracle ：文書内の少数派・多数派を既知として，多数派の全文をスタイルエンコーダに入力してスタイル表現を計算し，文書内の少数派全文を変換する．

4.4 評価尺度

本実験では，3.1 節の異常検知に対する評価と，3.2 節のスタイル変換に対する評価を行う．異なる 5 種類のシード値を設定して学習を行い，それらの平均値を各スコアとする．

異常検知の評価は文レベルで行う．各文に付与されたラベルについて, 外れ値ラベルを陽性とし, Precision, Recall, F_1 を計算する．

スタイル変換の評価は，出力文書のスタイル一貫性と入出力の同義性によって行い，それぞれスタイル判定器と BLEU スコア [25] を用いて計算する．前者に関しては，変換後の文がスタイル判定器によって文書中の多数派のスタイルであると分類されればスタイルの変換に成功したものとみなし，変換されなかった文も含めて入出力のすべての文を対象として正解率を計算することで，文書のスタイル一貫性に関するスコアとする．ただし，変換されなかった文に関しては，スタイル判定器を用いずに元のラベルをそのまま分類に利用する．

Yi らの実験に倣い，スタイル判定器は，4.1.2 節と同一のものを使用する．4.1.2 節で用いた GYAFC に対応する判定器と HSOL に対応する判定器に加えて，Sarcasm に対応するものと ShortJoke に対応する判定器を学習する．事前実験におけるこの判定器の分類精度は Sarcasm で 94.3%，GYAFC で 89.5%，ShortJoke で 96.9%，HSOL で 97.2%であり，スタイル変換の評価用擬似ラベルとしては十分な精度であると考えている．ただし，AmazonFormality に含まれる文に対して GYAFC で fine-tuning したスタイル判定器を用いて判定すると正解率は 65.2%であり，AmazonOffensiveness に含まれる文に対して HSOL で fine-tuning したスタイル判定器を用いて判定すると正解率は 87.8%であった．GYAFC は Q&A サービス由来のコーパス，HSOL は Twitter 由来のコーパスであり，レビュー由来の Amazon Review Data とはドメインが異なっていることがスコア低下の原因であると考えられる．このため，fine-tuning したスタイル判定器を AmazonFormality, AmazonOffensiveness に含まれる文を用いてさらに fine-tuning した．結果，AmazonFormality に対する正解率は 90.0%，AmazonOffensiveness に対する正解率は 97.0%となった．実際の文書による評価では，スタイル一貫性の評価にこちらのスタイル判定器を用いる．

後者の BLEU スコアについては，入力と出力の間で sacreBLEU (v2.2.0) [26] の BLEU.corpus_score を用いてスコアを計算し，内容が大きく変化していないか，すなわち入出力の同義性を評価する．この方法は多くの教師なしスタイル変換の先行研究で用いられており，self-BLEU とも呼ばれる [27], [28], [29]．一文単位の変換を行う既存のスタイル変換研究と異なり，本研究では変換されなかった文も含めて，入出力のすべての文を対象として計算し，文書単位での同義性スコアとして評価する点に注意されたい．

5. 結果と考察

5.1 少数派スタイルの文の検出

5.1.1 擬似文書データセットによる評価

表 3 に擬似文書に対する異常検知の Precision，Recall，F_1 を示す．学習時に用いたデータセットである Sarcasm, GYAFC, ShortJoke で F_1 を比較すると，提案手法（StyIns

表 4: 少数派スタイル文の検出性能（実文書）．小字は標準偏差．

少数派スタイル検知	AmazonFormality			AmazonOffensiveness		
	P	R	F_1	P	R	F_1
LOF	$0.301_{0.023}$	$\mathbf{0.177}_{0.066}$	$\mathbf{0.219}_{0.055}$	$0.044_{0.010}$	$\mathbf{0.179}_{0.051}$	$\mathbf{0.069}_{0.015}$
Transformer	$\mathbf{0.388}_{0.128}$	$0.012_{0.006}$	$0.023_{0.011}$	$\mathbf{0.070}_{0.050}$	$0.034_{0.027}$	$0.046_{0.035}$
Transformer（同時学習）	$0.330_{0.032}$	$0.052_{0.016}$	$0.089_{0.025}$	$0.051_{0.019}$	$0.079_{0.040}$	$0.061_{0.026}$

表 5: 文書のスタイル一貫性改善タスクの結果（擬似文書）．小字は標準偏差．

スタイル変換	少数派スタイル検知	スタイル一貫性				同義性 (self-BLEU)			
		Sarcasm	GYAFC	ShortJoke	HSOL	Sarcasm	GYAFC	ShortJoke	HSOL
無変換	検知なし	0.899	0.899	0.899	0.898	100.0	100.0	100.0	100.0
StyIns	検知なし	$0.605_{0.005}$	$0.677_{0.008}$	$0.877_{0.009}$	$0.867_{0.006}$	$90.6_{0.5}$	$89.6_{0.2}$	$80.7_{0.1}$	$78.1_{1.5}$
StyIns	LOF	$0.896_{0.008}$	$0.894_{0.011}$	$0.911_{0.007}$	$\mathbf{0.892}_{0.004}$	$94.5_{0.4}$	$92.1_{0.3}$	$\mathbf{86.1}_{0.3}$	$\mathbf{83.8}_{0.4}$
StyIns	Transformer	$0.938_{0.004}$	$0.919_{0.006}$	$0.940_{0.005}$	$0.891_{0.003}$	$\mathbf{95.1}_{0.6}$	$\mathbf{92.5}_{0.1}$	$84.3_{0.3}$	$83.7_{0.4}$
StyIns	Transformer（同時学習）	$\mathbf{0.943}_{0.005}$	$\mathbf{0.924}_{0.006}$	$\mathbf{0.954}_{0.005}$	$0.886_{0.003}$	$94.4_{0.6}$	$92.0_{0.3}$	$83.3_{0.8}$	$81.4_{0.6}$
StyIns	Oracle	$0.964_{0.003}$	$0.958_{0.004}$	$0.968_{0.004}$	$0.899_{0.003}$	$93.0_{0.4}$	$89.7_{0.4}$	$82.1_{0.4}$	$81.4_{0.3}$

+ Transformer（同時学習)) のスコアが最も高く，次いで StyIns + Transformer のスコアが高くなっている．Recall も同様の傾向を示している一方，Precision においては StyIns + Transformer (同時学習) と StyIns + Transformer とのスコアが逆転している．つまり，同時学習によって異常なスタイルの検出の取りこぼしは減ったものの，正しいスタイルも異常なものとして検出するようになっている．この理由として，少数派は数が少ないため，同時学習していないモデルはその偏りを学習して異常なスタイルを少なく見積もる傾向にあり，同時学習がそのバイアスを改善していると考えられる．学習時に用いなかった HSOL においては，スコアが他のデータセットのものを下回っており，改善の余地がある．

また，GYAFC におけるスコアは Sarcasm，ShortJoke のスコアと比べて低くなっている．この理由として，Sarcasm や ShortJoke に formal な表現と informal な表現が混在しており，それらに由来する擬似文書では，Formality の観点からは少数派としてラベル付けしなければならないような文も多数派としてラベル付けされていることから，Formality での少数派検出の学習が阻害されたと考えられる．

5.1.2 実際の文書による評価

表 4 に実際の文書に対する異常検知の Precision, Recall, F_1 を示す．学習時に用いたスタイルである Formality で F_1 を比較すると，Precision を除いて教師あり異常検知器 (StyIns + Transformer (同時学習), StyIns + Transformer) によるスコアが StyIns + LOF のスコアを下回っている．また**表 3** と比較すると，StyIns + Transformer（同時学習）と StyIns + Transformer の F_1 スコアがおよそ 10 分の 1 になっている．この理由としては，4.4 節で述べたような GYAFC と AmazonFormality のドメインの違いが原因であると考えられる．教師あり異常検知器が GYAFC の Q&A サービスのドメインに過剰に適応し，AmazonFormality のレビュードメインの表現に関して，教師なし異常検知器よりも判別能力が劣るようになったと思われる．

学習時に用いなかった Offensiveness においても，AmazonOffensiveness でのスコアが HSOL データセットでのスコアを下回り，教師なし異常検知器の方がスコアが高くなるという同様の傾向が見られる．この理由も，Formality と同様に HSOL のドメインと AmazonOffensiveness のドメインの違いが原因であると考える．

5.2 文書のスタイル一貫性改善

5.2.1 擬似文書データセットによる評価

表 5 に擬似文書に対するスタイル一貫性改善タスクの実験結果を示す．まず，スタイル一貫性スコアについて見ていくと，いずれのデータセットに対しても多数派と少数派が既知である StyIns + Oracle のスコアが最も高い．逆に最もスコアが低いのは StyIns + 検知なしで，一切変換を行わない無変換+ 検知なしよりも低くなっている．このことから，少数派スタイルの検出を行わずに文書中のすべての文のスタイルベクトルを用いてスタイルを平均化するような単純な手法では，却って文書スタイルの一貫性を損ねることがわかる．

少数派スタイルの文を検出するモデルの中では，学習内スタイルである Sarcasm，GYAFC，ShortJoke で提案手法である StyIns + Transformer（同時学習）が最も高い．これは，同時学習により少数派スタイルの文をより良く検出できたことで，より多くの変換すべき文をより適

表 6: 文書のスタイル一貫性改善タスクの結果（実文書）．小字は標準偏差．

スタイル変換	少数派スタイル検知	スタイル一貫性		同義性 (self-BLEU)	
		AmazonFormality	AmazonOffensiveness	AmazonFormality	AmazonOffensiveness
無変換	検知なし	**0.737**	0.969	100.0	100.0
StyIns	検知なし	$0.600_{0.006}$	$\mathbf{0.998}_{0.000}$	$92.7_{0.6}$	$92.2_{0.3}$
StyIns	LOF	$0.724_{0.006}$	$0.974_{0.002}$	$92.7_{0.6}$	$92.2_{0.3}$
StyIns	Transformer	$\mathbf{0.737}_{0.002}$	$0.970_{0.001}$	$\mathbf{93.8}_{0.2}$	$\mathbf{92.9}_{0.1}$
StyIns	Transformer （同時学習）	$0.736_{0.002}$	$0.971_{0.001}$	$92.7_{1.0}$	$91.9_{0.8}$
StyIns	Oracle	$0.960_{0.001}$	$0.999_{0.000}$	$92.8_{0.3}$	$93.2_{0.1}$

切なスタイルベクトルを参照して変換できたためだと考えられる．学習外スタイルの HSOL に関しては StyIns + LOF によるスコアが最も高い．HSOL で教師あり異常検知器を用いたモデルのスコアが低くなったのは，5.1.2 節で Amazon Review Data 由来のデータセットに対するスコアが低くなったのと同様に，学習時に用いた Sarcasm, GYAFC，ShortJoke のドメインやスタイルに過剰に適応してしまい，スタイルに関する汎化性能が教師なし異常検知器よりも低くなったからだと考えられる．また傾向として，表 3 と同様に GYAFC におけるスコアは Sarcasm, ShortJoke のスコアと比べて低くなっている．これも異常検知と同様に，Sarcasm や ShortJoke に混在する formal な表現と informal な表現が Formality の学習を阻害したからだと考えられる．

次に，同義性スコアについて見ていくと，Sarcasm と GYAFC で最もスコアが高いのは StyIns + Transformer であり，ShortJoke と HSOL では StyIns + LOF である．StyIns + Transformer（同時学習）のスコアは，どのデータセットに対しても，他の異常検知を行う手法と比較すると低くなっている．しかし多数派スタイルの文の同義性を破壊しにくい StyIns + Oracle とスコアが同等かそれ以上に高く，変換によって内容そのものが大きく変わってしまうことは少ないと考えられる．

5.2.2 実際の文書による評価

表 6 に実際の文書に対するスタイル一貫性改善タスクの実験結果を示す．まず，スタイル一貫性スコアについて見ていくと，いずれのデータセットに対しても多数派と少数派が既知である StyIns + Oracle のスコアが最も高い．逆にスコアが最も低いのは，AmazonFormality では表 5 と同様に StyIns + 検知なしであったが，AmazonOffensiveness では無変換 + 検知なしであった．

異常検知を行うモデルの中では，学習内スタイルである AmazonFormality に対しては教師あり異常検知を行う StyIns + Transformer と StyIns + Transformer（同時学習）のスコアが高く，学習外スタイルの AmazonOffensiveness に対しては教師なし異常検知を行う StyIns + LOF のスコアが最も高い．表 4 ではどちらのデータセットに対しても教師あり異常検知を行うモデルの Recall と F_1 が LOF よりも低かったにもかかわらず，AmazonFormality のスタイル一貫性スコアにおいては StyIns + LOF の方が低くなっている．スタイル一貫性においてスコアが逆転する理由は，スタイル参照文にノイズが混ざった場合のスタイル変換モジュールの性能の低さであると考えられる．なぜならば，StyIns + Oracle 以外で StyIns を用いるモデルでは，スコアが無変換 + 検知なしと同等か低下しているからである．

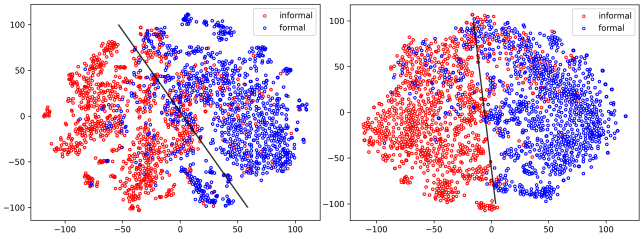
また表 5 と比較すると，異常検知を行う StyIns + Transformer（同時学習）と StyIns + Transformer，StyIns + LOF では，Formality のデータセットでのスコアは低下しているが，Offensiveness のデータセットでのスコアは寧ろ向上している．これはモデルの性能の向上ではなく，AmazonOffensiveness 内の文のスタイルが non-offensive に大きく偏っていたため，今回の評価方法では何も変換しなくてもスコアが高くなることが理由だと考察する．

次に，同義性スコアについて見ていくと，提案手法である StyIns + Transformer（同時学習）のスコアは，異常検知を行う StyIns + LOF や StyIns + Transformer と比較すると低い．しかし多数派スタイルの文の同義性を破壊しにくい StyIns + Oracle よりもスコアが高く，変換によって内容そのものが大きく変わってしまうことは少ないと考える．

しかし出力文を調査すると，“fit is just perfect , and installation had no problems .” という文が少数派となっていた文書では，異常検知に成功しているにもかかわらず出力が入力と全く同一の文になっており，変換に失敗していた．同様の失敗例は他にも見られた．このため，同義性スコアの高さは変換に失敗していることが原因であるとも考えられる．

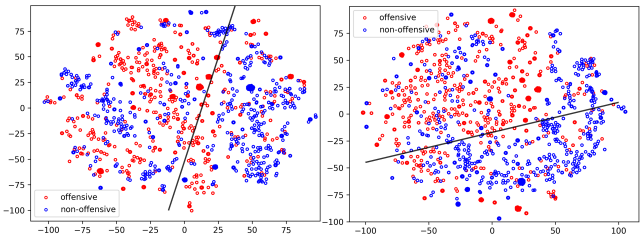
5.3 分析

同時学習を行わなかったモデル（StyIns + Transformer）と行ったモデル（StyIns + Transformer（同時学習））について，GYAFC データセットに対するスタイルエンコーダの出



(a) 同時学習なし　(b) 同時学習あり

図 3: GYAFC データセットに対してスタイルエンコーダ（異常検知との同時学習あり・なし）を適用して得られたスタイルベクトルの t-SNE による可視化.



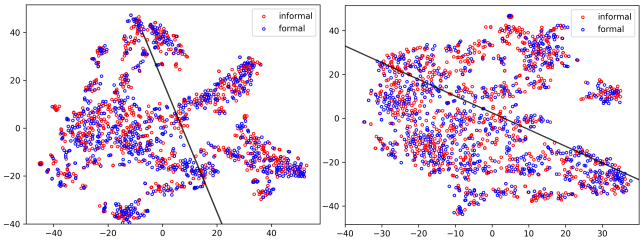
(a) 同時学習なし　(b) 同時学習あり

図 4: HSOL データセットに対してスタイルエンコーダ（異常検知との同時学習あり・なし）を適用して得られたスタイルベクトルの t-SNE による可視化.

力を t-SNE [30] で圧縮したプロットをそれぞれ示す（**図 3a**, **図 3b**）. 図中の黒線は，t-SNE で圧縮したデータ点を線形 Support Vector Machine (SVM) で分類した境界線である. SVM は scikit-learn (v1.1.1) の sklearn.svm.LinearSVC の実装を利用した. どちらの図においてもそれぞれのスタイル表現がクラスターを形成しているが，境界線を越えて逆のクラスターに含まれている点も見られる. SVM での分類の正解率は，同時学習なしの場合に 80.7%，同時学習ありの場合に 89.0%であり，同時学習ありの方が分類精度が高い. このため，同時学習がスタイルエンコーダでのスタイル分離に好ましい影響を与えたと言える.

StyIns + Transformer と StyIns + Transformer（同時学習） について，HSOL データセットに対するスタイルエンコーダの出力を t-SNE で圧縮したプロットをそれぞれ示す（**図 4a**, **図 4b**）. SVM での分類の正解率は，同時学習なしの場合に 61.0%，同時学習ありの場合に 66.5%であり，同時学習ありの方が分類精度が高いが，GYAFC での分類と比較すると低い. 目視でも境界線を越えている点が多く，各クラスタも疎であり，十分にスタイルの分離ができていないと言える. これが他のデータセットに対して著しくスコアが低い原因であると考えられる.

最後に，StyIns + Transformer と StyIns + Transformer（同時学習） について，AmazonFormality データセットに対するスタイルエンコーダの出力を t-SNE で圧縮したプロットをそれぞれ示す（**図 5a**, **図 5b**）. SVM での分類の



(a) 同時学習なし　(b) 同時学習あり

図 5: AmazonFormality データセットに対してスタイルエンコーダ（異常検知との同時学習あり・なし）を適用して得られたスタイルベクトルの t-SNE による可視化.

正解率は，同時学習なしの場合に 50.2%，同時学習ありの場合に 47.2%であり，ほぼランダム選択と精度が変わらない. 目視でも，どちらの図においてもクラスタの分離がされていないことがわかる. このため，Formality の特徴量が今回用いた擬似文書と実際の文書とで異なっており，Transformer に基づく教師あり異常検知器で後者の特徴量が捉えられていないと考察できる. なお，AmazonOffensiveness についても同様の結果であった.

6. おわりに

本研究では，文書のスタイル一貫性を改善することを目指し，Transformer に基づく教師あり異常検知モデルとスタイル変換モデルを同時学習する手法を提案した. 擬似文書を用いた評価では，学習内スタイルにおいて提案手法の同時学習が異常検知性能とスタイル一貫性改善性能を同時に向上させることを確認した. 一方で学習外スタイルでは教師なし手法よりも性能が低く，スタイルに関する汎化性能について改善の余地がある. 実際の文書を用いた評価では，異常検知性能において教師なし異常検知手法のスコアを下回り，スタイル一貫性は無変換よりも悪化した. これも提案手法の汎化性能の低さが原因であると考えられ，提案手法がスタイル一般の特徴を動的に獲得できているとは言い難い.

今後は未知ドメインへの対処のため，より多種類のスタイルを含むデータセットを用いることを検討する. 現状では 3 種類のスタイルのみで学習させているため，それ以外のスタイルはモデルにとって完全に未知である. しかしより多種類のスタイルを学習させることによって，スタイルの一般性を学習したり，入力の未知スタイルを既知のスタイルに近似して検知・変換したりすることが期待できる.

また異常検知の精度とスタイル変換性能を同時に向上させるため，スタイルエンコーダとして BERT などの事前学習済みモデルを用いることを検討する. 事前学習済みモデルは大規模データを用いて学習されており，多様なドメインに対して最大公約数的な言語表現を持っていることから，スタイル一貫性の改善タスクの性能が改善されること

が期待できる．

謝辞 この研究は国立情報学研究所（NII）CRIS と LINE 株式会社とが推進する NII CRIS 共同研究の助成を受けています．また，評価データのアノテーションにご協力くださった同研究室のメンバーに深く感謝申し上げます．

参考文献

[1] Wang, Y., Hoang, C. and Federico, M.: Towards Modeling the Style of Translators in Neural Machine Translation, *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, Association for Computational Linguistics, pp. 1193–1199 (online), DOI: 10.18653/v1/2021.naacl-main.94 (2021).

[2] Neff, G. and Nagy, P.: Automation, Algorithms, and Politics— Talking to Bots: Symbiotic Agency and the Case of Tay, *International Journal of Communication*, Vol. 10 (online), available from ⟨https://ijoc.org/index.php/ijoc/article/view/6277⟩ (2016).

[3] Qi, F., Chen, Y., Zhang, X., Li, M., Liu, Z. and Sun, M.: Mind the Style of Text! Adversarial and Backdoor Attacks Based on Text Style Transfer, *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pp. 4569–4580 (online), DOI: 10.18653/v1/2021.emnlp-main.374 (2021).

[4] Hovy, E.: Generating natural language under pragmatic constraints, *Journal of Pragmatics*, Vol. 11, No. 6, pp. 689–719 (1987).

[5] Prabhumoye, S., Tsvetkov, Y., Salakhutdinov, R. and Black, A. W.: Style Transfer Through Back-Translation, *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 866–876 (online), DOI: 10.18653/v1/P18-1080 (2018).

[6] Nogueira dos Santos, C., Melnyk, I. and Padhi, I.: Fighting Offensive Language on Social Media with Unsupervised Text Style Transfer, *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pp. 189–194 (online), DOI: 10.18653/v1/P18-2031 (2018).

[7] Cheng, Y., Gan, Z., Zhang, Y., Elachqar, O., Li, D. and Liu, J.: Contextual Text Style Transfer, *Findings of the Association for Computational Linguistics: EMNLP 2020*, pp. 2915–2924 (online), DOI: 10.18653/v1/2020.findings-emnlp.263 (2020).

[8] 京野長彦，吉永直樹，佐藤翔悦：異常検知に基づく文書のスタイル一貫性の改善，言語処理学会第 28 回年次大会 (NLP2022)，pp. 1396–1400 (2022).

[9] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L. u. and Polosukhin, I.: Attention is All you Need, *Advances in Neural Information Processing Systems*, Vol. 30, Curran Associates, Inc., (online), available from ⟨https://proceedings.neurips.cc/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf⟩ (2017).

[10] Mishra, A., Tater, T. and Sankaranarayanan, K.: A Modular Architecture for Unsupervised Sarcasm Generation, *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. 6144–6154 (online), DOI: 10.18653/v1/D19-1636 (2019).

[11] Rao, S. and Tetreault, J.: Dear Sir or Madam, May I Introduce the GYAFC Dataset: Corpus, Benchmarks and Metrics for Formality Style Transfer, *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pp. 129–140 (online), DOI: 10.18653/v1/N18-1012 (2018).

[12] Davidson, T., Warmsley, D., Macy, M. and Weber, I.: Automated Hate Speech Detection and the Problem of Offensive Language, *Proceedings of the International AAAI Conference on Web and Social Media*, Vol. 11, No. 1, pp. 512–515 (online), available from ⟨https://ojs.aaai.org/index.php/ICWSM/article/view/14955⟩ (2017).

[13] Ni, J., Li, J. and McAuley, J.: Justifying Recommendations using Distantly-Labeled Reviews and Fine-Grained Aspects, *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, Association for Computational Linguistics, pp. 188–197 (online), DOI: 10.18653/v1/D19-1018 (2019).

[14] Jhamtani, H., Gangal, V., Hovy, E. and Nyberg, E.: Shakespearizing Modern Language Using Copy-Enriched Sequence to Sequence Models, *Proceedings of the Workshop on Stylistic Variation*, pp. 10–19 (online), DOI: 10.18653/v1/W17-4902 (2017).

[15] Shen, T., Lei, T., Barzilay, R. and Jaakkola, T.: Style Transfer from Non-Parallel Text by Cross-Alignment, *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pp. 6833–6844 (online), available from ⟨https://proceedings.neurips.cc/paper/2017/file/2d2c8394e31101a261abf1784302bf75-Paper.pdf⟩ (2017).

[16] John, V., Mou, L., Bahuleyan, H. and Vechtomova, O.: Disentangled Representation Learning for Non-Parallel Text Style Transfer, *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 424–434 (online), DOI: 10.18653/v1/P19-1041 (2019).

[17] Breunig, M. M., Kriegel, H.-P., Ng, R. T. and Sander, J.: LOF: Identifying Density-Based Local Outliers, *ACM Sigmod Record*, Vol. 29, No. 2, pp. 93–104 (online), DOI: 10.1145/335191.335388 (2000).

[18] Zimek, A., Schubert, E. and Kriegel, H.-P.: A survey on unsupervised outlier detection in high-dimensional numerical data, *Statistical Analysis and Data Mining: The ASA Data Science Journal*, Vol. 5, No. 5, pp. 363–387 (online), DOI: 10.1002/sam.11161 (2012).

[19] Yi, X., Liu, Z., Li, W. and Sun, M.: Text Style Transfer via Learning Style Instance Supported Latent Space, *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI-20*, pp. 3801–3807 (online), DOI: 10.24963/ijcai.2020/526 (2020).

[20] Kang, D. and Hovy, E.: Style is NOT a single variable: Case Studies for Cross-Stylistic Language Understanding, *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pp. 2376–2387 (online), DOI: 10.18653/v1/2021.acl-long.185 (2021).

[21] Devlin, J., Chang, M.-W., Lee, K. and Toutanova, K.: BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding, *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Lan-*

guage Technologies, Volume 1 (Long and Short Papers), pp. 4171–4186 (online), DOI: 10.18653/v1/N19-1423 (2019).

[22] Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., Cistac, P., Rault, T., Louf, R., Funtowicz, M., Davison, J., Shleifer, S., von Platen, P., Ma, C., Jernite, Y., Plu, J., Xu, C., Scao, T. L., Gugger, S., Drame, M., Lhoest, Q. and Rush, A. M.: Transformers: State-of-the-Art Natural Language Processing, *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, (online), DOI: 10.18653/v1/2020.emnlp-demos.6 (2020).

[23] Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J. and Chintala, S.: PyTorch: An Imperative Style, High-Performance Deep Learning Library, *Advances in Neural Information Processing Systems 32* (Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E. and Garnett, R., eds.), Curran Associates, Inc., pp. 8024–8035 (online), available from ⟨http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf⟩ (2019).

[24] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M. and Duchesnay, E.: Scikit-learn: Machine Learning in Python, *Journal of Machine Learning Research*, Vol. 12, pp. 2825–2830 (2011).

[25] Papineni, K., Roukos, S., Ward, T. and Zhu, W.-J.: Bleu: a Method for Automatic Evaluation of Machine Translation, *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pp. 311–318 (online), DOI: 10.3115/1073083.1073135 (2002).

[26] Post, M.: A Call for Clarity in Reporting BLEU Scores, *Proceedings of the Third Conference on Machine Translation: Research Papers*, Association for Computational Linguistics, pp. 186–191 (online), DOI: 10.18653/v1/W18-6319 (2018).

[27] Cao, Y., Shui, R., Pan, L., Kan, M.-Y., Liu, Z. and Chua, T.-S.: Expertise Style Transfer: A New Task Towards Better Communication between Experts and Laymen, *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 1061–1071 (online), DOI: 10.18653/v1/2020.acl-main.100 (2020).

[28] Madaan, A., Setlur, A., Parekh, T., Poczos, B., Neubig, G., Yang, Y., Salakhutdinov, R., Black, A. W. and Prabhumoye, S.: Politeness Transfer: A Tag and Generate Approach, *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 1869–1881 (online), DOI: 10.18653/v1/2020.acl-main.169 (2020).

[29] Dai, N., Liang, J., Qiu, X. and Huang, X.: Style Transformer: Unpaired Text Style Transfer without Disentangled Latent Representation, *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, Florence, Italy, Association for Computational Linguistics, pp. 5997–6007 (online), DOI: 10.18653/v1/P19-1601 (2019).

[30] van der Maaten, L. and Hinton, G.: Visualizing Data using t-SNE, *Journal of Machine Learning Research*, Vol. 9, No. 86, pp. 2579–2605 (online), available from ⟨http://jmlr.org/papers/v9/vandermaaten08a.html⟩ (2008).