スタイル変換技術による対訳コーパスから同時通訳コーパスへの拡張

二又 航介[1,a)]　須藤 克仁[1,b)]　中村 哲[1,c)]

概要：同時通訳とは，入力文章が完結する前に目的言語の部分的な翻訳結果を訳出するタスクである．同時通訳システムを介したコミュニケーションでは，翻訳の遅延が円滑なコミュニケーションの大きな障害となるため，遅延を最小限にしつつ正確に部分訳出をする必要がある．特に英語と日本語のように語順が大きく異なる言語間の同時通訳では，訳出開始までの遅延が大きな問題となる．一方で，原言語の語順に近い形で訳出を行うことができれば，遅延を少なくすることができる．同時通訳システムの学習には通常，機械翻訳システムと同様に対訳コーパスが用いられる．同時通訳コーパスは，機械翻訳システムの学習に用いられる対訳コーパスと異なり，入力文が完結する前に目的言語の部分訳出を行った文から構成される対訳コーパスである．したがって，同時通訳システムの学習に用いられる対訳コーパスとして，同時通訳コーパスを用いることができれば，入力文を小さな部位に区切り逐次訳出できるため，訳出を終えるまでの遅延が少なくなる．しかし，現在利用可能な同時通訳コーパスの量は非常に少ないため，このような問題設定は現実的ではない．そこで本稿では，機械翻訳に用いられる対訳コーパスから，同時通訳コーパスへと拡張する手法について提案する．提案手法ではスタイル変換を用いることで，機械翻訳のスタイルから同時通訳へのスタイルへと変換を行う．また，スタイル変換により生成された疑似同時通訳文について現状での問題点について検討する．

1. はじめに

国際化に伴い，日常生活やビジネスの現場，国際会議などで外国語でのコミュニケーションの機会が増加している．外国語でのリアルタイムでの対話は，自分の母国語とは異なる言語でコミュニケーションを行う必要があるため，話し手，聞き手の双方に大きな負担をかける．そこで，利用者の負担を軽減しうる自動同時通訳のシステムが研究されている [3,8,17].

同時通訳とは，原言語の入力文の終了を待たずに目的言語への訳出を開始するものである．同時通訳システムを用いることで，講演や会話などをリアルタイムで理解する助けとなり，円滑なコミュニケーションを促進する．従来の機械翻訳システムでは，訳出を行う入力文における終端部が読み終わるまで訳出を行わない．講義や講演等の話し言葉では文が長くなる傾向があるため，文の終了を待って翻訳する枠組みでは翻訳結果が得られるまでの遅延が大きくなることが避けられない．しかし，同時通訳を用いたコミュニケーションでは，翻訳時の遅延が円滑なコミュニケーションの大きな障害となるため，遅延を最小限にしつつ正確に部分訳出をする必要がある．特に英語と日本語のように語順が大きく異なる言語間の同時通訳では，訳出開始までの遅延が大きな問題となる [19]．これは英語のような主要部先行 (head-initial) 言語と日本語のような主要部後続 (head-final) 言語の違いにより，訳出開始までの待ち時間が発生してしまうからである．つまり，被修飾部が先で修飾部が後に来るか (head-initial)，修飾部が先で被修飾部が後に来るか (head-final) の違いにより，待ち時間が発生する．図 1 に長い修飾部を持つ英文の日本語への訳出開始までに遅延が発生する例を示す．

図 1 は上から順に，英語の入力文，日本語の出力文，日本語への訳出タイミングを示す．入力文である'A brand-new computer on the desk which my father gave me on my birthday doesn't work now .' という英文では'A brand-new computer on the desk' が'my father gave me on my birthday' に対して修飾されている．この修飾部と被修飾部の関係により訳出開始までの遅延が発生する．

このように，主要部先行言語と主要部後続言語の間で比較的洗練された訳出を行うためには，多少の遅延が発生する．一方で，原言語の語順に近い形で訳出を行うことがで

[1] 奈良先端科学技術大学院大学
NAIST, Takayama-cho, Ikoma, Nara 630–0192, Japan
[a)] futamata.kosuke.fg6@is.naist.jp
[b)] sudoh@is.naist.jp
[c)] s-nakamura@is.naist.jp

A brand-new computer on the desk which **my father gave me** on my birthday doesn't work now .

父から 誕生日 に 貰った 、 机上 に ある 新しい コンピュータ は 今 故障 しています 。

(待ち時間...) 父から 誕生日 に 貰った 、 机上 に ある 新しい コンピュータ は . . .

図 1 訳出開始までに遅延が発生する例

A brand-new computer **on the desk** which my father gave me on my birthday doesn't work now .

机上 にある 新しい コンピュータ ですね、これは 父 から 誕生日 に 貰った もの です 、ですが 今 故障 しています 。

(待ち時間...) **机上 にある** 新しい コンピュータ ですね 、これは 父 から 誕生日 に 貰った もの です . . .

図 2 訳出開始までの遅延が少ない例

ければ，遅延を少なくすることができる．図 2 に遅延の少ない訳出例を示す．図 2 は図 1 と同様に上から英語の入力文，日本語の出力文，日本語への訳出タイミングを示す．英語の入力文は図 1 と同様のものであるが，日本語の出力文が異なる．図 1 の訳出例と比べ，図 2 の訳出例では語順の洗練性は無いが，助詞等により致命的な間違いはほとんど無く，聞き手の誤解は生じない．このような通訳方略を「順送り」[9] という．また，訳出開始までの遅延が図 1 の訳出例より少ない．以上の例のように，英日翻訳では最悪でも順送りを用いて訳出を行っても，決定的な問題は生じにくいと言える．これは，日本語が膠着語に分類され，文節における順序の入れ替えを比較的許容する言語であるからである．したがって，図 2 のように順送り方式で翻訳された文章から構成される同時通訳コーパスを用いて同時通訳システムを学習させれば，訳出時における遅延を最小限に留めることができると想定される．しかし，現在利用可能な同時通訳コーパスの量は非常に少ないため，このような問題設定は現実的ではない．そこで本研究では，同時通訳のような順送りの訳文を対訳コーパスにおける訳文から自動書き換えによって得る方法について検討する．

入力文を意味的に等価な文に書き換えるタスクの総称として，言い換え生成がある．言い換え生成には，テキストの難易度を制御するテキスト平易化 [12] や，テキストのスタイルを制御するスタイル変換 [1,14] などのタスクがある．スタイル変換では，変換前後の文ペアを必要とせず，スタイルを変換することが可能である．本研究では，通常の対訳コーパスの訳文と同時通訳コーパスの順送りの訳文がそれぞれ異なるスタイルであると仮定し，スタイル変換によって順送りの訳文への書き換えを行う．図 3 に英日対訳コーパスとスタイル変換により拡張される疑似英日同時通訳コーパスとの関係を示す．図 3 のオレンジ色の矩形で囲まれた部分は，英日対訳コーパスを表している．一方で，赤色の矩形で囲まれた部分は，英日対訳コーパスにおける入力文と，スタイル変換を適用することにより拡張された生成文を組み合わせることによって新たに作成された英日疑似同時通訳コーパスを表している．対訳コーパスの日本語文に対してスタイル変換を適用することにより，日本語の疑似同時通訳文を生成し，疑似同時通訳コーパスを作成する．

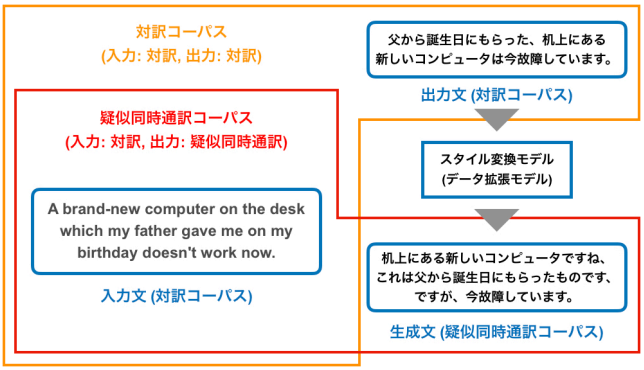

図 3 対訳コーパスと疑似同時通訳コーパスの関係

スタイル変換は，変更するスタイル情報が明確な場合に得に有効である．対訳コーパスと同時通訳コーパスには，スタイル情報を区別する様々な要因が関わってくるため，スタイル変換の手法をそのまま適用するだけでは不十分であると想定される．そこで，対訳コーパスにおける日本語文に事前並び替え [4,10] を適用することによって，同時通訳コーパスにおける日本語文との並び順が近くなるようにした．実験の結果，対訳コーパスにおける日本語文に事前並び替えを適用してスタイル変換を行うことで，生成文が短い単位に分割されるなど同時通訳文のスタイルに近くなることが明らかになった．

2. Style Transformer を用いたスタイル変換

スタイル変換とは，言い換え生成の一種である．この技術を利用することで，入力文における文体 (スタイル) を自動的に制御することができる．代表的なスタイル変換のタスクとしては，肯定的な文を否定的な文へ変換するものがある．スタイル変換を行う手法としては，意味情報とスタイル情報を分離し，分離後の意味情報に対してスタイル情報を付与する手法 [2,14] や，GAN や VAE により直接スタイルを制御する手法 [1,7,15] がある．

ここでスタイル変換の学習に用いるコーパスをそれぞれ $\{D_i\}_{i=1}^{K}$ と定義する．各コーパス D_i は特定のスタイルを含む文集合から構成される．英日対訳コーパスのスタイルから英日同時通訳コーパスのスタイルに変換するタスクを考えると，英日対訳コーパスにおける日本語文のスタイル D_1 および英日同時通訳コーパスにおける日本語文のスタイル D_2 の 2 つのデータセットから構成される．また，各 D_i に対応するスタイルを $S^{(i)}$ と定義する．このときスタイル変換の目的は，入力文 $\mathbf{x}$ と変換先スタイル $\hat{\mathbf{s}} \in \{\mathbf{s}^{(i)}\}_{i=1}^{K}$ が与えられたとき，入力文 $\mathbf{x}$ の意味情報を保持しつつ，スタイル情報 $\hat{\mathbf{s}}$ を含む文章 $\hat{\mathbf{x}}$ に変更することである．このような変換を可能にする関数 $f_\theta(\mathbf{x}, \mathbf{s})$ を学習させる．本研究では，Transformer と GAN を用いた Style Transformer [1] を使用した．次節以降では，背景知識として Style Transformer モデルの概要について説明する．

2.1 Style Transformer ネットワーク

Style Transformer ネットワークは Encoder-Decoder 構造を持つ Transformer [16] をベースにスタイル変数を組み込んだモデルである．入力文 (入力系列) $\mathbf{x}$，Encoder の出力である中間表現 $\mathbf{z}$，および出力文 (出力系列) $\mathbf{y}$ をそれぞれ以下のように定義する．

$$\mathbf{x} = \{x_1, x_2, ..., x_I\}$$
$$\mathbf{z} = \{z_1, z_2, ..., z_J\}$$
$$\mathbf{y} = \{y_1, y_2, ..., y_K\}$$

ここで，$x_i \in \mathbb{R}^{S\times 1}$ は i 番目の入力単語を表す one-hot ベクトル，I は入力文の長さ，$z_i \in \mathbb{R}^{T\times 1}$ は j 番目の Encoder の出力である中間表現ベクトル，J は中間ベクトルの長さ，$y_i \in \mathbb{R}^{U\times 1}$ は k 番目の出力単語を表す one-hot ベクトル，K は出力文の長さを表す．また，Transformer の Encoder および Decoder をそれぞれ，$Enc(\mathbf{x};\theta_E)$，$Dec(\mathbf{z};\theta_D)$ と定義する．θ_E および θ_D はそれぞれ Encoder と Decoder のパラメータを表す．

このとき，入力文 $\mathbf{x}$ に対して，Transformer の Encoder $Enc(\mathbf{x};\theta_E)$ は入力文を中間表現 $\mathbf{z}$ へ変換する．そして，Transformer の decoder $Dec(\mathbf{z};\theta_D)$ は中間表現 $\mathbf{z}$ から出力文 $\mathbf{y}$ の条件付き確率の積を式 1 のように求める．

$$p_\theta(\mathbf{y}|\mathbf{z}) = \prod_{t=1}^{m} p_\theta(y_t|\mathbf{z}, y_1, ..., y_{t-1}) \tag{1}$$

各ステップ t において，次の単語 $t+1$ に対する確率は softmax 分類器により，式 2 のように求められる．

$$p_\theta(y_t|\mathbf{z}, y_1, ..., y_{t-1}) = softmax(\mathbf{o}_t) \tag{2}$$

式 2 における o_t は Decoder の出力を表している．

通常の Transformer を用いて入力文のスタイル情報を制御するために，Style Transformer では語彙に対する単語埋め込みベクトルとは他に，スタイル情報を制御するための埋め込みベクトルの層を追加する．したがって，Style Transformer の Encoder $Enc(\mathbf{x}, \mathbf{s};\theta_E)$ では，入力文 $\mathbf{x}$ とスタイル情報 $\mathbf{s}$ を入力として受け取り，式 3 に示すように，入力文 $\mathbf{x}$ とスタイル情報 $\mathbf{s}$ の両方を条件とする確率を計算する．

$$p_\theta(\mathbf{y}|\mathbf{x}, \mathbf{s}) = \prod_{t=1}^{m} p_\theta(y_t|\mathbf{s}, \mathbf{z}, y_1, ..., y_{t-1}) \tag{3}$$

2.2 Discriminator ネットワーク

前節で述べたスタイル変換モデル $f_\mathbf{x}(\mathbf{x}, \hat{\mathbf{s}})$ を適用する際，一般的に変換元スタイルと変換先スタイルの対になったコーパスが存在することがまれであるため，スタイル変換モデルを教師あり学習の枠組みで学習させることは困難である．そこで，Style Transformer では Discriminator を用いて，独立した 2 つのコーパスから学習を行う．

Discriminator の学習時は，学習に用いるコーパスに含まれる文 $\mathbf{x}$ とその文と同一のスタイル情報 $\mathbf{s}$ を入力として再構築された文 $\mathbf{y} = f_\theta(\mathbf{x}, \mathbf{s})$ を正例として，異なるスタイル $\hat{\mathbf{s}}$ に変換された文 $\hat{\mathbf{y}} = f_\theta(\mathbf{x}, \hat{\mathbf{s}})$ を負例として識別するように Discriminator を学習させる．Discriminator の損失関数は式 4 に示すように，負の対数尤度を用いる．式 4 における $\mathbf{c}$ は正例または負例に分類されるクラスラベルを表す．

$$\mathcal{L}_{discriminator} = -p(\mathbf{c}|\mathbf{x}) \tag{4}$$

一方で Style Transformer の学習時には，Discriminator によって Style Transformer の出力 $\hat{\mathbf{y}} = f_\theta(\mathbf{x}, \hat{\mathbf{s}})$ が正例となる確率を最大化するように Style Transformer を学習させる．

2.3 Style Transformer + Discriminator

前節までで述べた Style Transformer ネットワークと Discriminator ネットワークを組み合わせることで，スタイル変換を行う．図 4 に Style Transformer 全体の学習過程を示す．図 4 に示すように，Style Transformer の学習対象は主に Self Reconstruction, Cycle Reconstruction, Style Controlling の 3 つに分けられる．また，入力として与えられるスタイルが入力文と同一のスタイル $\mathbf{s}$ の場合と，入力文のスタイルと異なるスタイル $\hat{\mathbf{s}}$ の場合によって，Self Reconstruction を学習対象にするのか，Cycle Reconstruction と Style Constrolling を学習対象にするのか異なる．

2.3.1 Self Reconstruction

入力として与えられるスタイルが入力文のスタイルと同一であるとき，入力文 $\mathbf{x}$ とスタイル $\mathbf{s}$ は同一コーパスからの入力であり，図 4 上部に示す灰色の矩形で囲まれた Self Reconstruction の損失関数を最小化する．この場合，モデ

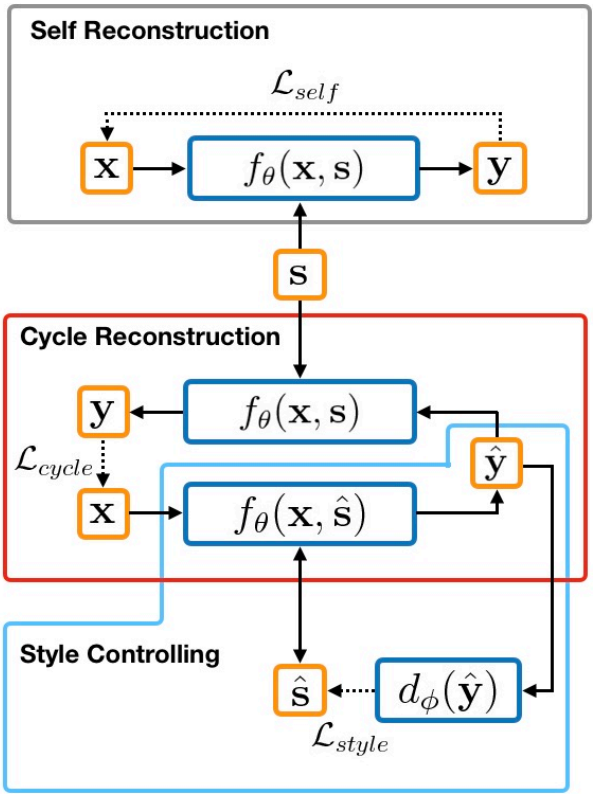

図 4　Style Transformer 全体の学習過程

ルは単に入力文を再構築するように学習を行い，式 5 に示す負の対数尤度を最小化する．

$$\mathcal{L}_{self}(\theta) = -p_\theta(\mathbf{y} = \mathbf{x}|\mathbf{x}, \mathbf{s}) \tag{5}$$

2.3.2　Cycle Reconsturction + Style Controlling

一方で，入力として与えられるスタイル情報が入力文のスタイルと異なるとき，入力文 $\mathbf{x}$ とスタイル $\hat{\mathbf{s}}$ は異なるコーパスからの入力であり，図 4 中部に示す赤色の矩形で囲まれた Cycle Reconsturction，下部に示す水色の矩形で囲まれた Style Controlling の 2 つの損失関数を最小化する．

Cycle Reconstruction では，入力文 $\mathbf{x}$ に含まれる意味情報を保持するために，異なるスタイル情報 $\hat{\mathbf{s}}$ を付与して生成された文 $\hat{\mathbf{y}} = f_\theta(\mathbf{x}, \hat{\mathbf{s}})$ に入力文のスタイル情報 $\mathbf{s}$ を付与して再び Style Transformer へ入力する．そして式 6 に示すように，負の対数尤度を最小化することで，元の入力文 $\mathbf{x}$ を再構築するように学習を行う．

$$\mathcal{L}_{cycle}(\theta) = -p_\theta(\mathbf{y} = \mathbf{x}|f_\theta(\mathbf{x}, \hat{\mathbf{s}}), \mathbf{s}) \tag{6}$$

Style Transformer が単に入力文 $\mathbf{x}$ を $\hat{\mathbf{y}} = f_\theta(\mathbf{x}, \hat{\mathbf{s}})$ により再構築するだけでは，モデルが入力を出力へと複製するだけになってしまう．そこで，図 4 に示す Style Controlling によって，Style Transformer の出力 $\hat{\mathbf{y}}$ を，Discriminator により正例として識別される確率を最大化するように Style Transformer の学習を行う．このとき Self reconstruction

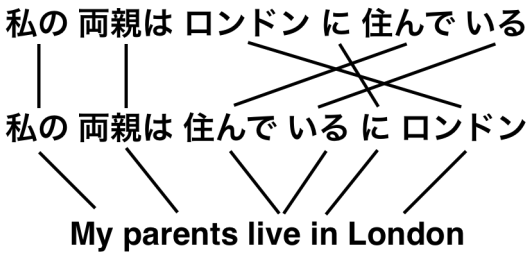

図 5　事前並び替えの適用例

と Cycle reconstruction と同様，式 7 に示す $\hat{\mathbf{s}}$ に関する負の対数尤度を最小化する．

$$\mathcal{L}_{style} = -p(\mathbf{c} = \hat{\mathbf{s}}|f_\theta(\mathbf{x}, \hat{\mathbf{s}})) \tag{7}$$

3.　事前並び替え

事前並び替えは統計的機械翻訳 (SMT) で使用される手法である．SMT において，翻訳言語間の語順の相違は翻訳精度に大きな影響を与えることが知られており [4]，この問題を解決する手法として，翻訳機に入力する前に原言語における文の語順を目的言語における文の語順に近づくように並び替える事前並び替えが提案されている [4, 10]．特に日英翻訳のように単語の語順が大きく異る言語対において，事前並び替えは SMT で精度を改善することが示されている．図 5 に事前並び替えの適用例を示す．図 5 の上から順に，事前並び替えを適用する前の日本語文，事前並び替えを適用した後の日本語文，事前並び替えを適用するための英語参照文である．「私の 両親は ロンドン に 住んで いる」という日本語文が，事前並び替えを適用することによって，「私の 両親は 住んで いる に ロンドン」という語順に変わる．事前並び替えを適用した日本語文は英語参照文と比較すると，単語間の交差が無くなっていることがわかる．

3.1　事前並び替えのスタイル変換への適用

前節で述べたスタイル変換の手法は，変更するスタイル情報が明確な場合に特に有効である．なぜなら既存のスタイル変換手法では，局所的な単語を置き換えたり，削除したりと表現力が限定されるからである．例えば，肯定的な文を否定的な文へ変換する場合などが挙げられる．

本研究の目的である対訳コーパスにおける日本語文から疑似同時通訳文へとスタイル変換を行うタスクでは，ある箇所で文を区切ったり，単語間の入れ替えなどが発生するため，既存のスタイル変換モデルでは表現力に欠けると想定される．そこで，本研究ではスタイル変換の学習時に事前並び替えを適用した文を用いる．

事前並び替えを英日対訳コーパスにおける日本語文に適用することによって，参照文である英日対訳コーパスの英文との単語間の交差が少なくなると想定される．また，英

日同時通訳コーパスにおける日本語文は第一章で述べたように，英語から順送り方式で翻訳されていると仮定すると，英日同時通訳コーパスにおける英文と日本語文における単語間の交差は比較的少ないと想定される．したがって，英日対訳コーパスにおける日本語文に事前並び替えを適用することによって，英日同時通訳コーパスにおける日本語文との単語間との交差も少なくなると推測される．

4. 実験

Style Transformer によるスタイル変換を英日対訳コーパスにおける日本語文から英日同時通訳コーパスの日本語文へのデータ拡張へ適用する実験を行い，現状での問題点について検討した．また英日対訳コーパスにおける日本語文に対しては，事前並び替えを適用する場合と適用しない場合の 2 通りの実験を行った．

4.1 実験設定

モデルの実装には Pytorch を用いた．また，Style Transformer の Encoder と Decoder, および Discriminator は共に 4 層とし，単語埋め込みベクトルや隠れ状態ベクトルの次元数は 256, ミニバッチのサイズは 128，語彙のサイズは 16000 とした．最適化アルゴリズムには Adam [5] を使用し，gradient clipping は 5 にして学習を行った．また評価指標には，スタイル変換の自動評価尺度として一般的に使用されている 2 値分類器による正解率, BLEU [13], Perplexity を使用した．2 値分類器による正解率により，英日対訳コーパスにおける日本語文が英日同時通訳コーパスにおける日本語文のスタイルに変換されたかどうか評価し，BLEU スコアにより，意味情報を保持しているかどうかを評価する．また，Perplexity により，スタイル変換が適用された文が流暢なものであるかどうかを評価する．

4.2 データセット

英日対訳コーパスにおける日本語文のスタイルから英日同時通訳の日本語文におけるスタイルへとスタイルを変換する実験を行うにあたって，日本語話し言葉コーパス (CSJ) [18] と独自に収集した TED コーパスを使用した．CSJ は日本語で行われた講演の音声を集めたコーパスであるため，その書き起こし文を対訳コーパスにおける日本語文と仮定して使用した．TED コーパスは英語で行われた講演を日本語へ同時通訳したものである．事前並び替えには，Nakagawa [10] の Bracketing Transduction Grammar(BTG) による手法を用いた．また，事前並び替えの学習には対訳コーパスが必要であるため，ASPEC [11] を使用した．ASPEC は中規模のコーパスで，比較的長文かつ専門用語が多く複雑な文章から構成されている．表 1 に各コーパスの詳細を示す．

入力単位は形態素とし，Mecab [6] を用いて単語分割を行った．また CSJ と TED に関しては，文の長さが 30 トークンを超えるものと，その対訳文を学習データから削除するフィルタリングを行った．

表 1　CSJ，TED，ASPEC のコーパスサイズ

Corpus	Number of sentences		
	Train	Val	Test
CSJ	3509	500	500
TED	28606	500	500
ASPEC	1000000	1790	1812

表 2　自動評価による実験結果

	ACC	BLEU	PPL
CSJ-TED	89.0	48.8	73.5
CSJ(pre-ordered)-TED	96.0	21.8	361.5
CSJ-CSJ(preordered)	30.2	28.3	242.7

4.3 自動評価による実験

スタイル変換を適用するコーパスのペアとして，「CSJ から TED(CSJ-TED)」，「事前並び替えを適用した CSJ から TED(CSJ(preordered)-TED)」，「CSJ から事前並び替えを適用した CSJ(CSJ-CSJ(preordered))」の計 3 種類を対象とした．CSJ-CSJ(preordered) に関しては．事前並び替えを適用した CSJ の文が英語参照文との単語間交差が少なくなり，同時通訳らしい文の並び順になることを期待している．

2 値分類器によるスタイルの正解率 (ACC)，BLEU スコア，Perplexity(PPL) による評価結果を表 2 に示す．CSJ-TED，CSJ(preordered)-TED の ACC の評価値は高かった．BLEU, PPL の評価値に関しては CSJ-TED のスコアは高いものの，CSJ(preordered)-TED のスコアは低かった．一方で，CSJ-CSJ(preorrdered) では，ACC, BLEU, PPL の全ての値で CSJ-TED, CSJ(preordered)-TED と比較して低かった．

4.4 自動評価に関する考察

表 3 および表 4 にそれぞれ CSJ-TED, CSJ(preordered)-TED のスタイル変換例を示す．入力例はスタイル変換を適用する対象の文，変換例は入力文に対してスタイル変換を適用した文，原文は入力文に事前並び替えを適用する前のものをそれぞれ表す．表 3 の Exmple(1) では語尾が「ました」が「ます」に変換された．Example(2) では「私」が「僕」に変換された．一方で表 4 の Example(1) では「学習 データ は 、 です」が「学習 データ は **ない** です 、」に変換された．Example(2) では，文頭に「私」が追加された．

以上の結果からわかるように，CSJ-TED のスタイル変換例では単に一部の単語が置換されるだけであり，本研究で期待するような同時通訳のスタイルが現れることがなかった．BLEU スコアと PPL の値は比較的高かったが，

置換された単語がごく一部に限られ，自然で流暢な文が生成されたためであると想定される．また ACC に関しては，事前学習させた 2 値分類器によるスタイルの判断基準が期待通りのものにならなかったためであると想定される．

CSJ(preordered)-TED のスタイル変換例でも同様に同時通訳のスタイルが現れることがなかった．しかし事前並び替えを適用したことにより，スタイル変換された文で語順の変化や，新たに単語が追加されるなどの傾向がみられた．その結果として流暢さが損なわれ BLEU と PPL のスコアが低くなったと想定される．

表 5 に CSJ-CSJ(preordered) のスタイル変換例を示す．表 5 における参照文は入力文に対して事前並び替えを適用したものを表す．表 5 の Example(1) および Example(2) では文の途中に「です」,「ます」が新たに追加された．Example(3) では文頭に「ついに」，語尾付近に「きちんと」が追加された．

以上の結果から CSJ-CSJ(preordered) のスタイル変換例では，長い文を短い文単位に区切る傾向があることが分かった．Example(1) の参照文のように，事前並び替えによって文が短い単位で区切られる例がみられた．この結果，変換例に関しても長い文を短い文に区切るように変換されたと想定される．しかし一方で Example(3) のように，文としての流暢さが損なわれている変換例も数多くみられた．その結果 BLEU スコアと PPL の値が低くなったと想定される．ACC に関しては，CSJ-TED のスタイル変換例と同様に，事前学習させた 2 値分類器によるスタイルの判断基準が期待通りにならなかったためであると想定される．

4.5 人手評価による実験

前節で示したように，2 値分類器による正解率，BLEU スコア，Perplexity などの自動評価指標のみを用いて，同時通訳のスタイルが適用された文を評価するのは困難である．そこでスタイル変換によって生成された文が同時通訳らしい文であるか人手評価を行った．前節の表 3，表 4，表 5 に示したように，自動評価では CSJ-CSJ(preordered) の性能が一番低かったが，長い文を短い文に区切る傾向がみられるなど，最も同時通訳のスタイルに近い文が生成された．その結果を踏まえて，CSJ-CSJ(preordered) のペアを対象として人手評価を行った．人手評価には 7 人の被験者に，スタイル変換前の文 (参照文) とスタイル変換後の文 (生成文) を提示し，次に挙げる 3 つの指標を満たすかどうかを 5 段階で評価してもらった．1 つめの指標は「参照文と比較し、生成文は短い単位に区切られているかどうか (Segmentation)」であり，これは生成文が同時通訳らしいものであるか計測する．2 つめの指標は「生成文は日本語として自然で流暢であるかどうか (Fluency)」であり，これは生成文が日本語として正しいものであるか計測する．3 つめの指標は「参照文と生成文が意味的に同一であるかどうか (Identity)」であり，これは参照文と生成文の意味情報が変わっていないか計測する．

人手評価による実験結果の平均値を表 6 に示す．表 6 における'Segmentation'，'Fluency'，'Identity' のそれぞれの値は被験者 7 人の平均値を示す．'All samples' は全 50 サンプルの平均値，Segmentation $\geq$ 3.0 は'Segmentation' の平均値が 3.0 以上であったサンプル，Fluency $\geq$ 3.0 は'Fluency' の平均値が 3.0 以上であったサンプル，そして Identity $\geq$ 3.0 は'Identity' の平均値が 3.0 以上であったサンプルの平均値をそれぞれ表す．

表 6 の実験結果に示すように，全サンプルから計算された評価平均では，'Fluency' の平均値が'Segmentation' と'Identity' より低かった．また，表 6 の 2 行目以降に示すように，それぞれの評価指標における平均値が 3.0 以上のサンプルのみを用いて分析を行った．サンプル数はそれぞれ Segmentation $\geq$ 3.0 では 30，Fluency $\geq$ 3.0 では 10，Identity $\geq$ 3.0 では 28 である．分析の結果，Segmentation $\geq$ 3.0 のサンプルでは，'Fluency' の平均値が下がり，Fluency $\geq$ 3.0 のサンプルでは，'Segmentation' の平均値が大きく下がり，'Identity' の平均値が大きく上がった．'Identity' $\geq$ 3.0 のサンプルでは，'Fluency' の平均値が上がった．

4.6 人手評価に関する考察

表 6 の結果から，'Segmentation' の平均値が上がると，'Fluency' の平均値が下がり，'Fluency' の平均値が上がると，'Segmentation' の平均値が下がることが伺える．また，'Identity' の平均値が上がると，'Fluency' の平均値が上がった．しかし，Fluency $\geq$ 3.0 と Identity $\geq$ 3.0 の間に'Identity' の平均値に大きな違いはみられなかった．したがって，'Segmentation' と'Fluency' の間および'Segmentation' と'Identity' の間にはトレードオフの関係性があるのではないかと推測される．つまり，スタイル変換された文が短い単位に区切られると，文が流暢ではなくなり，入力文と変換文の意味が異なってしまう傾向にあり，短い単位に区切られなければ，流暢で意味的に同一な文が生成される傾向にある．表 7 に「'Segmentation' の平均値が特に高く，'Fluency' と'Identity' の平均値が特に低い変換例」を，表 8 に「'Segmentation' の平均値が特に低く，'Fluency' と'Identity' の平均値が特に高い変換例」を示す．

表 7 における Example(1) の変換例 (CSJ(preordered)) では，文が短い単位に区切られているものの，「発音 が 高い ところ **ます**」など日本語として不自然であったり，「最も 低い **事故** へ」など，入力文 (CSJ) と意味が異なる箇所がみられる．Example(2) も同様に変換例 (CSJ(preordered)) が短い単位に区切られているものの，日本語として不自然であり，意味を理解することが難しい．このように'Segmentation' の平均値が高く'Fluency' と'Identity' の平均値が低

表 3　CSJ-TED のスタイル変換例

Example(1)	
入力文 (CSJ)	世界 戦争 が ヨーロッパ から 始まり ました 。
変換例 (TED)	世界 戦争 が ヨーロッパ から 始まり **ます** 。

Example(2)	
入力文 (CSJ)	私 が 言わ なくちゃ いけ ない 内容 に 入る 前 に ， 少し 私 の 自己 紹介 です 。
変換例 (TED)	私 が 言わ なくちゃ いけ ない 内容 に 入る 前 に ， 少し **僕** の 自己 紹介 です 。

表 4　CSJ(preordered)-TED のスタイル変換例

Example(1)	
原文 (CSJ)	学習 データ は 、 こちら と 同じ もの です 。
入力文 (CSJ(preordered))	学習 データ は 、です 同じ もの と こちら 。
変換例 (TED)	学習 データ は **ない** です 、 同じ もの と こちら 。

Example(2)	
原文 (CSJ)	一方 アジア 人 は 、日本人 学 部 生 は 、 僕 一人 で 少ない 方 でし た 。
入力文 (CSJ(preordered))	た 一方 アジア 人 は 、 日本人 学 部 生 は 、 でし 僕 一人 で 方 少ない 。
変換例 (TED)	**私** 一方 アジア 人 は 、 日本人 学 部 生 、 一人 で 少ない 。

い変換例では，文が短い単位に区切られたものの，文が流暢ではなくなり，入力文と意味が異なってしまう傾向がみられた．一方で表 8 における Example(1) および Example(2) の変換例 (CSJ(preordered)) では，日本語として自然であり，入力文 (CSJ) と意味が同一であるが，短い単位に区切られることがなかった．

’Segmentation’ の平均値が上がると’Fluency’ および’Identity’ の平均値が下がる要因としては，事前並び替えの結果に一貫性がないことが考えられる．Kawara et al. [4] では，ルールベースのように一貫性のある並び替えを行うことが、ニューラル機械翻訳の精度を向上するためには重要であることが示されている．本研究では，機械学習による事前並び替え手法の一つである BTG を用いたが，事前並び替えの結果に多くの間違えがみられ，一貫性が担保されていないため，’Fluency’ と’Identity’ の平均値が低くなったと想定される．したがって，ルールベース基づく事前並び替えなどの手法を用いることで，事前並び替えの結果に一貫性を担保すれば，’Fluency’ と’Identity’ の評価値を向上させることが可能なのではないかと推測される．

5. おわりに

本論文では，スタイル変換を対訳コーパスに適用することで，同時通訳コーパスへと拡張する手法について提案した．提案手法では，スタイル変換と事前並び替えを組み合わせることにより，疑似同時通訳文を生成した．実験の結果，スタイル変換により文を短い単位で分割することが可能であり，同時通訳らしい文章を一定の割合で生成できることが明らかになった．しかし，実際の同時通訳文は，単に文を短い単位に分割するだけではなく，単語の語順が大きく入れ替わったり，前後の文脈における繋がりを明確にするため，単語を追加するなど様々な要因が関わってくる．今後は変換文の自然らしさ，意味的同一性を担保しつつ，より同時通訳らしい文を生成する手法について検討したい．スタイル変換によって生成された疑似同時通訳文を実際に同時通訳システムの学習データとして使用することで，翻訳精度を向上させることを目指す．

また，本研究では自動評価指標として，スタイル変換手法の評価に一般的に用いられる 2 値分類器による正解率, BLEU スコア, Perplexity を使用したが，疑似同時通訳文としての妥当性を測る尺度については今後さらに検討が必要である．事前並び替えの結果とスタイル変換によって生成された疑似同時通訳文における単語の交差距離を計算するなど，新たな評価指標を導入する必要がある．

6. 謝辞

本研究の一部は JSPS 科研費 (JP17H06101) の助成を受けたものである．

参考文献

[1] 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), available from ⟨https://www.aclweb.org/anthology/P19-1601⟩ (2019).

[2] Fu, Z., Tan, X., Peng, N., Zhao, D. and Yan, R.: Style Transfer in Text: Exploration and Evaluation, *CoRR*, Vol. abs/1711.06861 (online), available from ⟨http://arxiv.org/abs/1711.06861⟩ (2017).

[3] Gu, J., Neubig, G., Cho, K. and Li, V. O. K.: Learning to Translate in Real-time with Neural Machine Translation, *CoRR*, Vol. abs/1610.00388 (online), available from ⟨http://arxiv.org/abs/1610.00388⟩ (2016).

表 5 CSJ-CSJ(preordered) のスタイル変換例

Example(1)	
入力文 (CSJ)	講演 音声 認識 の 識別 率 は 、 今 の ところ 七十 パーセント 程度 です 。
参照文 (CSJ)	識別 率 の 講演 音声 認識 は 、 です 程度 ところ の 今 七十 パーセント 。
変換例 (CSJ(preordered))	講演 音声 認識 の 識別 率 は **です** 今 の ところ 七十 パーセント 程度 。

Example(2)	
入力文 (CSJ)	次 に 実際 に 決定 木 を 構築 し て の 選択 し 検索 を 行う 実験 と いう の を 行ない まし た 。
参照文 (CSJ)	ます し は 、 を 使用 この 三 種類 いう と 動詞 目的 語 名詞 し と 関係 する 使用 て 。
変換例 (CSJ(preordered))	次 に 実際 に 決定 木 を 構築 し **ます** 選択 し 検索 を 行ない **ます** 実験 と いう の を 行ない まし た 。

Example(3)	
入力文 (CSJ)	次 に 本 研究 に おけ ます システム の 概要 を 説明 いたし ます 。
参照文 (CSJ)	ます いたし 次 に 概要 の おけ ます に 本 研究 システム を 説明 。
変換例 (CSJ(preordered))	**つい に** 本 研究 に おけ ます システム の 概要 を **きちんと** 説明 **て** 。

表 6 人手評価による実験結果の平均値

	Num samples	Segmentation	Fluency	Identity
All samples	50	3.03	2.42	3.11
Segmentation $\geq$ 3.0	30	3.72	2.02	2.95
Fluency $\geq$ 3.0	10	2.37	4.00	3.74
Identity $\geq$ 3.0	28	2.96	2.76	3.73

表 7 'Segmentation' の平均値が特に高く'Fluency' と'Identity' の平均値が特に低い変換例

Example(1)	
入力文 (CSJ)	第 四 声 は 、発音 が 高い ところ から 最も 低い ところ へ 急激 に 移行 し て いき ます 。
参照文 (CSJ)	ます いき し 第 四 声 に 急激 移行 へ ところ は 最も 低い から ところ 、 発音 が 高い て 。
変換例 (CSJ(preordered))	四 声 は 、 発音 が 高い ところ **ます** 、 最も 低い **事故** へ 急激 に 移行 し て 。

Example(2)	
入力文 (CSJ)	大体 この くらい で 終わり に したい と 思い ます 。
参照文 (CSJ)	ます 大体 に 終わり したい と 思い で この くらい 。
変換例 (CSJ(preordered))	**ます** 、 **思い** くらい で 、 終わり に したい と 思い **よく** 。

表 8 'Segmentation' の平均値が特に低く'Fluency' と'Identity' の平均値が特に高い変換例

Example(1)	
入力文 (CSJ)	特に ですね 自転車 を 使った 旅 が 好き です 。
参照文 (CSJ(preordered))	です 特に 旅 を 使っ ね です 自転車 た が 好き 。
変換例 (CSJ(preordered))	特に ですね 自転車 を 使った 旅 が 好き 。

Example(2)	
入力文 (CSJ)	で この 町 の 中心 で ある 公園 に ついて 御 説明 し たい と 思い ます 。
参照文 (CSJ(preordered))	で ます 思い と たい し で ついて 御 説明 に 公園 ある 中心 の この 町 。
変換例 (CSJ(preordered))	で この 町 の 中心 で ある 公園 に ついて 御 説明 し たい と 思い ます 。

[4] Kawara, Y., Chu, C. and Arase, Y.: Recursive Neural Network Based Preordering for English-to-Japanese Machine Translation, *Proceedings of ACL 2018, Student Research Workshop*, Melbourne, Australia, Association for Computational Linguistics, pp. 21–27 (online), DOI: 10.18653/v1/P18-3004 (2018).

[5] Kingma, D. P. and Ba, J.: Adam: A Method for Stochastic Optimization, *CoRR*, Vol. abs/1412.6980 (2014).

[6] Kudo, T.: Mecab: Yet another part-of-speech and morphological analyzer.

[7] Lample, G., Subramanian, S., Smith, E., Denoyer, L., Ranzato, M. and Boureau, Y.-L.: Multiple-Attribute Text Rewriting, *International Conference on Learning Representations*, (online), available from ⟨https://openreview.net/forum?id=H1g2NhC5KQ⟩ (2019).

[8] Ma, M., Huang, L., Xiong, H., Liu, K., Zhang, C., He, Z., Liu, H., Li, X. and Wang, H.: STACL: Simultaneous Translation with Integrated Anticipation and Controllable Latency, *CoRR*, Vol. abs/1810.08398 (online), available from ⟨http://arxiv.org/abs/1810.08398⟩ (2018).

[9] 水野的：同時通訳の理論—認知的制約と訳出方略，朝日出版社 (2015).

[10] Nakagawa, T.: Efficient Top-Down BTG Parsing for Machine Translation Preordering, *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, Beijing, China, Association for Computational Linguistics, pp. 208–218 (online), DOI: 10.3115/v1/P15-1021 (2015).

[11] Nakazawa, T., Yaguchi, M., Uchimoto, K., Utiyama, M., Sumita, E., Kurohashi, S. and Isahara, H.: ASPEC: Asian Scientific Paper Excerpt Corpus, *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*, Portorož, Slovenia, European Language Resources Association (ELRA), pp. 2204–2208 (online), available from ⟨https://www.aclweb.org/anthology/L16-1350⟩ (2016).

[12] Nishihara, D., Kajiwara, T. and Arase, Y.: Controllable Text Simplification with Lexical Constraint Loss, *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics: Student Research Workshop*, Florence, Italy, Association for Computational Linguistics, pp. 260–266 (online), available from ⟨https://www.aclweb.org/anthology/P19-2036⟩ (2019).

[13] 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*, Philadelphia, Pennsylvania, USA, Association for Computational Linguistics, pp. 311–318 (online), DOI: 10.3115/1073083.1073135 (2002).

[14] 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)*, Melbourne, Australia, Association for Computational Linguistics, pp. 866–876 (online), DOI: 10.18653/v1/P18-1080 (2018).

[15] Shen, T., Lei, T., Barzilay, R. and Jaakkola, T. S.: Style Transfer from Non-Parallel Text by Cross-Alignment, *CoRR*, Vol. abs/1705.09655 (online), available from ⟨http://arxiv.org/abs/1705.09655⟩ (2017).

[16] 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 30* (Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S. and Garnett, R., eds.), Curran Associates, Inc., pp. 5998–6008 (online), available from ⟨http://papers.nips.cc/paper/7181-attention-is-all-you-need.pdf⟩ (2017).

[17] Yanagita, T., Sakti, S. and Nakamura, S.: Incremental TTS for Japanese Language, *Proc. Interspeech 2018*, pp. 902–906 (online), DOI: 10.21437/Interspeech.2018-1561 (2018).

[18] 小磯花絵，前川喜久雄：『日本語話し言葉コーパス』の設計の概要と書き起こし基準について，一般社団法人情報処理学会，pp. 41–48 (2011).

[19] 千葉紀子：日英同時通訳コーパスを用いた連体修飾節の訳出方略に関する研究 (2013).