Vol.2019-NL-241 No.20
2019/8/30

同時翻訳のためのConnectionist Temporal Classificationを用いたニューラル機械翻訳

帖佐 克己[1,a)] 須藤 克仁[1,b)] 中村 哲[1,c)]

概要：同時翻訳は文の入力が終了する前にその文の翻訳を開始するタスクである．このタスクでは翻訳精度と訳出までの遅延時間がトレードオフの関係にあり，システムを構築する際には翻訳を行うタイミングを適切に決定する必要がある．本研究では，ニューラル機械翻訳においてこの訳出タイミングを適応的に決定する方法を提案する．提案手法では目的言語側の語彙に訳出を行わない代わりに出力するためのメタトークン‘<wait>’を追加し，損失関数として Connectionist Temporal Classification（CTC）と呼ばれるアルゴリズムを目的関数に導入する．CTC によって 縮約すると正解系列と一致するような‘<wait>’を含む系列全て に対して最適化を行うことで翻訳モデルと訳出タイミング制御を同時に最適化することができ，さらに訳出タイミングを適応的に決定することも可能となる．また，このモデルを英語から日本語への同時翻訳タスクに対して適用し，その翻訳結果の精度や問題点について検討する．

1. はじめに

同時翻訳は文の入力が終了する前にその文の翻訳を行うタスクである．同時翻訳は話し言葉による講演や会話などに求められる発話理解のリアルタイム性を実現する助けとなり，円滑なコミュニケーションを推進することができる．

従来の機械翻訳システムでは文の終端が入力されるまで翻訳を行わない．しかし，話し言葉ではしばしば文境界が明らかでない場合があり，文を統一的な処理単位としようとすると文同定の不整合が生じやすい．この結果として複数文が結合されたものや不完全な文が翻訳器への入力として与えられる場合があり，文単位かつ文の終端まで入力されることを仮定している従来の機械翻訳システムでは学習時と異なった環境で翻訳を行うこととなるために精度の減少を招く．これらの問題に対して，従来の自動同時翻訳手法では文を小さいチャンクに分割して翻訳することにより翻訳結果が得られるまでの遅延時間を削減する試みが行われてきた [1]．しかし，逆に遅延を小さくすればするほど訳出の際に参照できる文内文脈が限られてくるため，翻訳精度が下がってしまうという問題も発生する．このことから，同時翻訳システムを構築する際には翻訳を行う単位を適切に決定し，遅延と翻訳精度との間のトレードオフを調節する必要がある．また，英語と日本語のような語順が大きく異なる言語対での翻訳は特に遅延が大きくなる傾向にあるため，自動同時翻訳において翻訳精度を落とすことなく遅延を小さくすることが難しい．

これら問題を解決するニューラル機械翻訳（Neural Machine Translation; NMT）モデル [2, 3] としていくつかの手法が提案されている．Gu ら [4] は既存の翻訳モデルに対して 1 単語を入力する *READ* と 1 単語を訳出する *WRITE* の 2 つのアクションを定義し，各タイムステップにおいてモデルがどちらのアクションを行うべきであるかを決定する分類器を強化学習によって学習する手法を提案している．この手法は一定の翻訳精度を保ったまま遅延を削減することに成功しているが，翻訳器が文の部分的な情報から翻訳することに対して最適化されていないという問題がある．また，Ma ら [5] は “Wait-k”モデルと呼ばれる非常にシンプルなモデルが提案している．このモデルは原言語側の文の入力に対して常に k トークン遅れた状態で翻訳文の生成を行う．k 単語のみの遅延では訳出に必要な語句がまだ入力されていない状況が起こり得るが，そうした場合でも強制的に訳語を予測し生成することになる．この方法により翻訳を行う機構と単語の予測を行う機構の両方を統合して扱うことが可能になり，それを End-to-End で学習することができる．この手法は非常にシンプルにも関わらず英語からドイツ語，中国語から英語の同時翻訳タスクにおいて高い精度を達成している．また，k を変化させることで遅延の大きさを直感的に調整することができるという利

1 奈良先端科学技術大学院大学
Nara Institute of Science and Technology
a) k-chousa@is.naist.jp
b) sudoh@is.naist.jp
c) s-nakamura@is.naist.jp

点もある．しかし，この手法では遅延を適応的に調節することが出来ないため，英語と日本語のような語順が大きく異なる言語対では 1 つのフレーズの長さが k よりも大きい場合などに翻訳を失敗してしまうということが我々の以前の研究によって判明している [6].

これらの問題に対して，本研究では目的言語側の語彙に新たなメタトークン<wait>を追加し，訳出するべきでないタイミングではモデルがそのメタトークンを出力することで訳出タイミングを適応的に決定する手法を提案する．また，<wait>をどのタイミングで出力すべきであるかという正解データが無いという問題を解決するために Connectionist Temporal Classification (CTC)[7] と呼ばれるアルゴリズムを損失関数として導入する．CTC は発話音声の音素予測のような出力タイミングが不定な系列のモデリングなどに用いられる誤差関数であり，この誤差関数を用いることによって<wait>を含む正解系列に対して最適化を行うことが可能となる．さらに，英語から日本語への同時通訳タスクに対してこの手法を適用し，その翻訳結果の精度や問題点について分析・議論する．提案手法により訳出タイミングを適応的に決定することで，日本語と英語のような語順が大きく異なる言語対における同時翻訳タスクで一定の遅延を保ったまま翻訳精度を向上させることが期待できる．

提案手法の評価を行うため，英語から日本語への自動同時翻訳タスクでの実験を行った．実験には比較的短い文で構成されている small_parallel_enja と長い文で構成されている ASPEC の 2 種類のコーパスを用いた．実験結果より，提案手法は訳出タイミングを適応的に決定することが可能であり，比較的短い文においては非常に小さな遅延で一定の翻訳精度を実現できることが分かった．

2. “Wait-k”モデルによる同時翻訳

はじめに，背景知識として NMT による機械翻訳モデル [2, 3] および “Wait-k”モデル [5] について説明する．

原言語文（入力系列）X および目的言語文（出力系列）Y を以下のように定義する．

$$X = \{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_I\},$$
$$Y = \{\mathbf{y}_1, \mathbf{y}_2, ..., \mathbf{y}_J\}.$$

ここで，$\mathbf{x}_i \in \mathbb{R}^{S\times 1}$ 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，$\mathbf{y}_j \in \mathbb{R}^{T\times 1}$ は j 番目の出力単語を表す one-hot ベクトル，J は出力文の長さを表す．

このとき，原言語から目的言語への翻訳という問題は以下の文に対する条件付き確率を最大化する目的言語文 $\hat{Y}$ を求めることによって解くことができる．

$$\hat{Y} = \arg\max_{Y} p(Y|X) \tag{1}$$

一般に，この条件付き確率は原言語文 X と時刻 j までに生成した目的言語文 $\mathbf{y}_{<j}$ から単語 $\mathbf{y}_j$ に対する条件付き確率の積の形として分解される．従来の NMT モデルにおいては式（2）のように分解されるのに対して，“Wait-k”モデルでは文の先頭のみが入力された状態から訳出を行う必要があることから式（3）のように定義される．

$$p(Y|X) = \prod_{j=1}^{J} p_\theta(\mathbf{y}_j|\mathbf{y}_{<j}, X) \tag{2}$$

$$p(Y|X) = \prod_{j=1}^{J} p_\theta(\mathbf{y}_j|\mathbf{y}_{<j}, \mathbf{x}_{<g(j)}) \tag{3}$$

ここで，$\mathbf{x}_{<g(j)}$ は時刻 $g(j)$ までに入力された入力文を表す．また，$g(j)$ は Decoder が時刻 j までトークンを生成したときに Encoder によって処理されているトークン数を表し，以下のように定義される．

$$g(j) = \begin{cases} k + j - 1 & (j < I - k) \\ I & (otherwise) \end{cases} \tag{4}$$

このとき，k は目的言語文の生成が原言語文の入力よりも k トークン遅延していることを表すパラメータであり，“Wait-k”モデルでは固定の値を取る．

モデルは Encoder (§2.1) と Attention+Decoder (§2.2) の 2 つの機構から構成され，そのどちらも Recurrent Neural Network(RNN) を用いて構成される．

2.1 Encoder

Encoder は原言語文 X を入力として受け取り，RNN を通じて順方向の隠れ状態ベクトル $\overrightarrow{\mathbf{h}_i} (1 \le i \le I)$ を返す．

$$\overrightarrow{\mathbf{h}_i} = RNN(\overrightarrow{\mathbf{h}_{i-1}}, \mathbf{x}_i). \tag{5}$$

従来の機械翻訳モデルでは入力文を逆順に並べたものを同様に入力することで逆方向の隠れ状態ベクトルを計算するが，同時通訳タスクでは文末が確定しない状況で文を処理する必要があるため，順方向のベクトルのみを利用することとなる．

2.2 Attention+Decoder

Attention+Decoder では Encoder で計算された入力文の隠れ状態ベクトルから翻訳文の単語を 1 つずつ生成する．Decoder の RNN は隠れ状態と過去の出力系列から自己再帰的に単語を生成する．出力単語 $\mathbf{y}_j$ の生成確率 $p_\theta(\mathbf{y}_j|\mathbf{y}_{<j}, \mathbf{x}_{\le g(j)})$ は以下のように定義される．

$$p_\theta(\mathbf{y}_j|\mathbf{y}_{<j}, \mathbf{x}_{\le g(j)}) = softmax(\mathbf{W}_s \tilde{\mathbf{b}_j}), \tag{6}$$
$$\tilde{\mathbf{b}_j} = tanh(\mathbf{W}_c[\mathbf{c}_j; \mathbf{d}_j]), \tag{7}$$
$$\mathbf{d}_j = RNN(\mathbf{d}_{j-1}, \mathbf{y}_{j-1}). \tag{8}$$

ここで，$\mathbf{W}_c, \mathbf{W}_p$ は学習されるパラメータである．また，$\mathbf{c}_j$ は文脈ベクトルである．この $\mathbf{c}_j$ を求めるために Attention

と呼ばれる機構を用いる．Attention 機構では，入力文の各隠れ状態ベクトル $\overrightarrow{\mathbf{h}_i}$ に対応する時間ステップ j における重み α_{ij} を計算し，その重みとその隠れ状態ベクトルとの重み付き平均を取ることで $\mathbf{c}_j$ が以下のように求められる．

$$\mathbf{c}_j = \sum_{t=1}^{g(j)} \alpha_{ij} \overrightarrow{\mathbf{h}_i}, \tag{9}$$

$$\alpha_{ij} = \frac{exp(\mathbf{d}_j^T \overrightarrow{\mathbf{h}_i})}{\sum_{t'=1}^{g(j)} exp(\mathbf{d}_j^T \overrightarrow{\mathbf{h}_{t'}})}. \tag{10}$$

3. CTC を用いた同時翻訳モデル

本研究では，目的言語側の語彙にメタトークン<wait>を追加し，訳出することのできないタイミングではモデルが<wait>を出力することにより訳出タイミングを適応的に決定する手法を提案する．

本節では，提案手法を実現するために用いた 2 つの誤差関数について説明する．はじめに機械翻訳における一般的な誤差関数である Softmax Cross-Entropy について説明し（§3.1），その後 Connectionist Temporal Classification (§3.2)，Delay Penalty (§3.3) について説明する．

3.1 Softmax Cross-Entropy

Softmax Cross-Entropy(SCE) は機械翻訳などの多クラス分類問題に対して一般的に用いられる誤差関数であり，以下のように定義される．

$$\ell_{ent} = -\sum_{j=1}^{J}\sum_{k=1}^{K} \mathbf{y}_{jk} \log p_\theta(\mathbf{y}_{jk}|\mathbf{y}_{<j}, \mathbf{x}_{<g(j)}). \tag{11}$$

ここで $\mathbf{y}_{jk}$ は目的言語文の j 番目の単語に対応する 1-hot ベクトルの k 番目の要素を表し，$p(\mathbf{y}_{jk}|\cdot)$ は $\mathbf{y}_{jk}$ の生成確率を表す．

SCE を NMT の誤差関数として用いる場合には出力系列に対して 1 対 1 で対応付けられる正解系列が必要となるが，提案手法においては<wait>をどのタイミングで訳出するべきであるかが明らかでないため，SCE を提案手法にシンプルに適用することが出来ない．この問題に対して本研究では，<wait>を出力可能なタイムステップ t $(t \leq g(I))$，すなわち原言語文が入力されている間に<wait>が出力された場合にはトークンに対する誤差を 0 とすることとした．

3.2 Connectionist Temporal Classification

前述の SCE では<wait>を訳出する正解タイミングがわからないことから<wait>にかかる誤差を 0 としたため，<wait>の生成に関しては学習が行われないという問題がある．これに対して，我々は Connectionist Temporal Classification(CTC)[7] と呼ばれる誤差関数を用いて系列単位での学習を行うことにより解決を試みた．

CTC では，<wait>などのトークンを挿入，もしくは各記号が連続して出力されることを許すことにより，パス $\boldsymbol{\pi}$ と呼ばれる出力の系列を長さ T に拡張する．パス $\boldsymbol{\pi}$ は全てのトークンの繰り返しと<wait>を消去することによって元の出力系列 $\mathbf{y} = \Omega^{-1}(\mathbf{y})$ が復元される．CTC の損失関数は $\boldsymbol{\pi} \in \Omega(\mathbf{y})$ の全てのパスの確率の和として以下のように定義され，forward-backward アルゴリズムによって計算される．

$$\begin{aligned}\ell_{ctc} &= \sum_{\boldsymbol{\pi}\in\Omega(\mathbf{y})} p(\boldsymbol{\pi}|X) \\ &= \sum_{\boldsymbol{\pi}\in\Omega(\mathbf{y})} \prod_{t=1}^{T} p(\pi_t|\pi_{<t}, \mathbf{x}_{g(t)}).\end{aligned} \tag{12}$$

ここで，π_t は $\boldsymbol{\pi}$ の時刻 t の出力である．

3.3 Delay Penalty

さらに，直接的に遅延の大きさを調節するために我々は Delay Penalty を導入した．Delay Penalty は以下のように定義され，出力されたトークンが遅延を発生させる場合，すなわち<wait>または直前と同じトークンが出力された場合にのみ計算を行った．

$$\ell_{del} = \sum_{j=1}^{J} \log(1 - w_j) \tag{13}$$

$$w_j = p(<wait>|\mathbf{y}_{<j}, \mathbf{x}_{<g(j)}) + p(\mathbf{y}_{j-1}|\mathbf{y}_{<j}, \mathbf{x}_{<g(j)}) \tag{14}$$

3.4 Loss Function

本研究では，これまで紹介した 3 つの誤差関数を組み合わせた以下の誤差関数を最適化の際に用いた．

$$\ell = \ell_{ent} + \ell_{ctc} + \alpha\ell_{del} \tag{15}$$

ここで α は遅延の大きさを調整するためのハイパーパラメータである．

4. 実験

提案モデルの評価を行うために英語から日本語への同時翻訳タスクでの実験を行い，その翻訳結果の精度や問題点について検討した．

4.1 実験設定

モデルの実装には PyTorch[*1] を用いた．また，Encoder と Decoder の RNN はそれぞれ 2 層の単方向 LSTM[8] とし，input feeding[2] を行った．単語埋め込みベクトルや隠れ状態ベクトルの次元数は 512 とし，ミニバッチのサイズは 64 とした．最適化アルゴリズムには Adam[9] を使用し，learning rate は 10^{-3}，gradient clipping は 50 に設定して学習を行った．ドロップアウトの確率 p は 0.3 とし，

[*1] https://pytorch.org

表 1　実験に用いたコーパスに関する情報.

Corpus	Number of Sentence		
	Train	Valid.	Test
small_parallel_enja	50k	500	500
ASPEC	964k	1790	1812

learning rate には各 epoch ごとに validation loss が減少しない場合にのみ $1/\sqrt{2}$ を掛けることによって減衰を行った．また，テストは最も小さい validation loss を記録したモデルによって行った．

英語から日本語への同時翻訳タスクでの実験を行うにあたり，パラレルコーパスとして small_parallel_enja*2 および ASPEC[10] を使用した．この small_parallel_enja は小規模なコーパスで，4 単語から 16 単語までの一般的なドメインの文から構成されている．また，ASPEC は中規模のコーパスで，比較的長文で専門用語が多いなどの特徴がある複雑な文章から構成されている．表 1にコーパスの詳細情報を示す．

英語および日本語の入力単位はサブワード [11, 12] とし，Sentencepiece*3 を用いてトークナイズを行った．語彙は原言語と目的言語でそれぞれ別に用意し，語彙サイズは small_parallel_enja では 4000, ASPEC では 8000 としてそれぞれ作成を行った．また，文の長さが 60 トークンを超えるもの，文の長さの比が 9 を超える対訳ペアに関しては，そのペアを学習データから削除を行った．

ベースラインには単方向 LSTM による Attention-based Encoder-Decoder による全文からの翻訳（Full sentence）および “Wait-k” モデルを用いた．翻訳精度の評価尺度には，機械翻訳の自動評価尺度として一般的に使用されている BLEU[13] および RIBES[14] を使用した．評価の際のトークナイズには kytea[15] を用いた．

4.2　実験 1: small_parallel_enja

初めに，比較的短い文でのモデルの性能を確認するために small_parallel_enja をコーパスとして用いて実験を行った．“Wait-k” モデルの遅延トークン数 k は 3 および 5，提案手法のハイパーパラメータ $\alpha = \{0, 0.01, 0.03, 0.05\}$ に設定した．

small_parallel_enja での実験における自動評価尺度および遅延の大きさによる評価結果を表 2に示す．Full sentence のスコアをモデルの取りうる評価の上界だと考えると，提案手法はほぼ同じ性能で小さな遅延を実現できていることがわかる．また，平均遅延が同程度の “Wait-k” モデルと提案手法を比較すると，少し精度が減少しているもしくは同程度の精度が得られていることがわかる．この精度の減少は，“Wait-k” モデルでの遅延が固定で分散が 0 であるのに対して提案手法では訳出タイミングを適応的に決定できる

*2 https://github.com/odashi/small_parallel_enja
*3 https://github.com/google/sentencepiece

表 2　small_parallel_enja における自動評価尺度および遅延の大きさによる評価結果．遅延はトークン数の平均および標準偏差で表されている．

モデル		遅延	BLEU	RIBES
Full sentence [2]		9.75 (± 2.69)	34.53	84.03
Wait-k [5]	k=3	3.00 (± 0.00)	31.06	82.46
	k=5	5.00 (± 0.00)	33.29	83.45
Ours	α=0.00	4.32 (± 3.14)	28.01	81.78
	α=0.01	4.29 (± 3.16)	30.42	82.60
	α=0.03	2.88 (± 2.95)	26.47	80.51
	α=0.05	0.80 (± 1.96)	22.60	77.86

ことから，遅延の分散が生じて遅延が少ない文での翻訳精度が減少しているのではないかと考えられる．

4.3　実験 2: ASPEC

次に，長文かつ 1 つのフレーズが非常に長いものとなる状況でのモデルの性能を確認するため，ASPEC をコーパスとして用いて実験を行った．“Wait-k” モデルの遅延トークン数 k は 5 および 7，提案手法のハイパーパラメータ $\alpha = \{0.03, 0.05, 0.1\}$ に設定した．

表 3　ASPEC における自動評価尺度および遅延の大きさによる評価結果. 遅延はトークン数の平均及び標準偏差で表されている．

モデル		遅延	BLEU	RIBES
Full sentence [2]		29.81 (± 14.30)	32.22	80.17
Wait-k [5]	k=5	5.00 (± 0.00)	21.53	71.40
	k=7	7.00 (± 0.00)	23.20	73.21
Ours	α=0.03	23.03 (± 14.08)	24.86	72.59
	α=0.05	21.96 (± 13.88)	22.45	70.60
	α=0.1	17.13 (± 12.69)	23.66	72.27

ASPEC での実験における自動評価尺度および遅延の大きさによる評価結果を表 3に示す．実験結果より提案手法での遅延が “Wait-k” モデルと比較して非常に大きいものになっていることがわかる．これは，ASPEC のドメインが科学技術論文でありデータ中に長いフレーズが多く含まれていることが原因ではないかと考えられる．名詞句などの長いフレーズを訳す際にはフレーズ全体を入力される必要があることが多いが，“Wait-k” モデルでは遅延の大きさが固定なので予測をするなどして無理やりにでも訳出を行うために遅延を小さく保つことができる．それに対して，提案手法では CTC によってフレーズ全体が入力されるまで遅延を発生させるパスも探索可能であるため，その結果 <wait>などがが出力されやすくなり遅延が大きくなるのではないかと考えられる．

4.4　考察

以上の実験結果より，提案手法は長文では遅延が大きくなる傾向にあるが，訳出タイミングを適応的に決定するこ

とが可能であり，文全体を入力するよりも小さな遅延において一定の翻訳精度を実現できることがわかった．

表 4 に small_parallel_enja における翻訳結果の例を示す．Example (1) では，提案手法で訳出タイミングを適応的に決定することにより正しく翻訳出来ている例を示している．この例では原言語文で文末に入力される”swimm ing” という単語が参照訳では「水泳」という単語として非常に早い段階で訳出されている．先行研究である “Wait-k”モデルでは遅延の大きさが固定であるため，この例のような語順の入れ替えの幅が k を超える場合にうまく訳出することができなくなってしまうので間違った翻訳を出力してしまう．それに対して，提案手法では”swimm”が入力されるまで<wait>を出力して訳出タイミングを調整することによって，「泳ぐ」という単語を訳出することが出来ている．

一方で，提案手法の翻訳結果には Example (2) のような文の入力まで<wait>を出力し続けるようなものが多く見られる．この例では，遅くともピリオドが入力されたタイミングで何かしらの訳出が行えるはずであるのにも関わらず<wait> を出力している．これは，英語・日本語のような語順が大きく異なる言語対では遅延を出力することに対するペナルティよりも全文から翻訳を行って SCE や CTC での誤差を小さくする方が誤差関数全体としては誤差が小さくなるようなことがあるからではないかと考えられる．特に，現状学習や評価に用いているコーパスは一般の機械翻訳タスクに用いられているものであり同時翻訳に最適化されたものでない．そのため，人間の同時通訳者の翻訳文などに見られる原言語文と語順が大きく変わらない参照訳を学習データとして学習を行うことでこの問題を緩和できるのではないかと考えられる．

5. まとめ

本研究では，NMT を用いた自動同時翻訳において語順の大きく異なる言語間では必須となる適応的な訳出タイミングの決定を行うことができる方法を提案した．提案手法では NMT の目的言語側の語彙に新たなメタトークン <wait>を追加し，訳出を行わない代わりに<wait>を出力することで訳出タイミングの決定を行う．また，従来のトークン単位で誤差を計算する Softmax Cross-Entropy に加えて系列単位で誤差を計算することのできる Connectionist Temporal Classification を用いることによって，正解データのない<wait>の出力タイミングに対して最適化を行うことを試みた．また，提案モデルを英語から日本語への自動同時翻訳タスクに対して適応し，その翻訳結果の精度や問題点について分析・議論を行った．その結果，比較的短い文に対する自動同時翻訳に関しては先行研究と同程度の精度を達成し，また訳出タイミングを適応的に決定できていることがわかった．また，長い文に対する自動同時翻訳に関しても，遅延が大きいという問題が残るが，一定の翻訳精度を達成した．

今後の課題としては，翻訳結果から遅延の大きさ別に翻訳精度を計算するなどの分析を行うことや遅延をトークン数ではなく実際の音声入力の時間で評価する，原言語文との語順が大きく変化しないデータでの学習などが考えられる．

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] Tomoki Fujita, Graham Neubig, Sakriani Sakti, Tomoki Toda, and Satoshi Nakamura. Simple, lexicalized choice of translation timing for simultaneous speech translation. In InterSpeech, pages 3487–3491, Lyon, France, August 2013.

[2] Thang Luong, Hieu Pham, and Christopher D. Manning. Effective approaches to attention-based neural machine translation. In Proceedings of EMNLP, pages 1412–1421, September 2015.

[3] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. arXiv preprint arXiv:1409.0473, 2014.

[4] Jiatao Gu, Graham Neubig, Kyunghyun Cho, and Victor OK Li. Learning to translate in real-time with neural machine translation. In Proceedings of EACL, volume 1, pages 1053–1062, 2017.

[5] Mingbo Ma, Liang Huang, Hao Xiong, Kaibo Liu, Chuanqiang Zhang, Zhongjun He, Hairong Liu, Xing Li, and Haifeng Wang. Stacl: Simultaneous translation with integrated anticipation and controllable latency. arXiv preprint arXiv:1810.08398, 2018.

[6] 帖佐 克己, 須藤 克仁, and 中村 哲. 英日同時通訳におけるニューラル機械翻訳の検討. In 言語処理学会第 25 回年次大会, 3 月 2019.

[7] Alex Graves, Santiago Fernández, Faustino Gomez, and Jürgen Schmidhuber. Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks. In Proceedings of the 23rd international conference on Machine learning, pages 369–376. ACM, 2006.

[8] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. Neural computation, 9(8):1735–1780, 1997.

[9] Diederik P. Kingma and Jimmy Lei Ba. Adam: a method for stochastic optimization. In Proceedings of ICLR2016, 2015.

[10] Toshiaki Nakazawa, Manabu Yaguchi, Kiyotaka Uchimoto, Masao Utiyama, Eiichiro Sumita, Sadao Kurohashi, and Hitoshi Isahara. Aspec: Asian scientific paper excerpt corpus. In Proceedings of LREC 2016, pages 2204–2208, Portoro, Slovenia, may 2016.

[11] Rico Sennrich, Barry Haddow, and Alexandra Birch. Neural Machine Translation of Rare Words with Subword

表 4 small_parallel_enja での翻訳例．<w>は遅延を表す．

Example (1)

原言語文:	he	did	n	’	t	care	for	swimm	ing	.		
参照訳:	彼	は	水泳	が	得意	で	は	な	かっ	た	。	
Wait-k (k=3):	<w>	<w>	<w>	彼	は	野球	を	飲	み	ま	せ	ん で し た 。
Ours (α=0.03):	彼	は	<w>	<w>	<w>	<w>	<w>	泳	ぐ	の	が	好 き で は な かっ た 。

Example (2)

原言語文:	it	’	s	business	.	
参照訳:	それ	が	仕事	で	す	。
Ours (α=0.03):	<w>	<w>	<w>	<w>	<w>	それ は 商売 で す 。

Units. In Proceedings of ACL, pages 1715–1725, Berlin, Germany, August 2016.

[12] Taku Kudo and John Richardson. SentencePiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. In Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations, pages 66–71, Brussels, Belgium, November 2018. Association for Computational Linguistics. DOI: 10.18653/v1/D18-2012. URL https://www.aclweb.org/anthology/D18-2012.

[13] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. In Proceedings of ACL, pages 311–318, Philadelphia, Pennsylvania, USA, July 2002.

[14] Hideki Isozaki, Tsutomu Hirao, Kevin Duh, Katsuhito Sudoh, and Hajime Tsukada. Automatic evaluation of translation quality for distant language pairs. In Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing, pages 944–952, Cambridge, MA, October 2010. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/D10-1092.

[15] Graham Neubig, Yosuke Nakata, and Shinsuke Mori. Pointwise prediction for robust, adaptable Japanese morphological analysis. In Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies, pages 529–533, Portland, Oregon, USA, June 2011. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/P11-2093.