二値符号予測と誤り訂正に基づく コンパクトなニューラルネットワーク翻訳モデル

小田 悠介 † Philip Arthur† Graham Neubig‡† 吉野 幸一郎 †§ 中村 哲 †

† 奈良先端科学技術大学院大学 ‡Carnegie Mellon University § 科学技術振興機構

{oda.yusuke.on9, philip.arthur.om0}@is.naist.jp, gneubig@cs.cmu.edu, {koichiro, s-nakamura}@is.naist.jp

1 はじめに

ニューラルネットワークに基づく翻訳モデル (Neural Machine Translation: NMT) [1] では, 出力単語の生成確率を単語ごとに個別に予測するため, ネットワークの出力層で大きな空間計算量が必要となる. 本研究では, 出力層を単語と対応する二値符号を予測するモデルに置き換えることで, 空間計算量を大幅に削減する手法を提案する. 提案法により, 理想的な場合で出力層の大きさが従来法の対数程度まで削減可能となる. これに加え, 本研究では提案法の翻訳精度を向上させる手法として, 単語の出現頻度に基づく従来法との混合モデル, 及び誤り訂正符号を用いて出力層の頑健性を向上させる手法についても導入する. 実験により, これらを組み合わせた翻訳モデルにより従来法に匹敵する翻訳精度を達成可能であることを示す.

2 二値符号予測を用いた翻訳モデル

2.1 出力層の二値符号化

図 1 は Encoder-Decoder 及び Attention と呼ばれる構造に基づく標準的な NMT モデル [1] の概略である. ここで, 機械翻訳の入出力は単語などの離散値であり, ネットワークがこれらを扱うには何らかの方法で離散値とベクトルの相互変換を行う必要がある. 頻繁に使用されるのは 1-Hot 表現であり, これは単語 w に対応する番号 $\mathrm{id}(w) \in \{1..V\}$ の要素のみが 1, 他は 0 となるような疎ベクトル $\boldsymbol{\delta}_{\mathrm{id}(w)} \in \mathbb{R}^V$ で単語を表現する手法である. ここで V は翻訳モデルが扱う語彙数である. 図 1 の出力層では $\boldsymbol{\delta}_{\mathrm{id}(w)}$ を正解単語の確率分布と見なし, ネットワークにより生成された Softmax 確率 $\boldsymbol{v} \in \mathbb{R}^V$ との交差エントロピー

$$\begin{aligned} L_{\mathcal{H}}(\boldsymbol{v}, \mathrm{id}(w)) &:= \mathcal{H}(\boldsymbol{\delta}_{\mathrm{id}(w)}, \boldsymbol{v}) \\ &= -u_{\mathrm{id}(w)} + \log \operatorname{sum} \exp \boldsymbol{u} \end{aligned}$$

を最小化するようネットワークを最適化する. ただし

$$\boldsymbol{v} := \exp \boldsymbol{u} / \operatorname{sum} \exp \boldsymbol{u}, \quad \boldsymbol{u} := W_{hu}\boldsymbol{h} + \boldsymbol{\beta}_u$$

である. ここで $\operatorname{sum} \boldsymbol{x}$ はベクトル $\boldsymbol{x}$ の要素の総和, x_i は i 番目の要素とし, $W_{hu} \in \mathbb{R}^{V \times H}, \boldsymbol{\beta}_u \in \mathbb{R}^V$ は学習可能なパラメータ, H は隠れ層の大きさである.

このモデルは W_{hu} の定義から明らかに, パラメータに対して $O(HV)$ の空間計算量が必要となる. 典型的

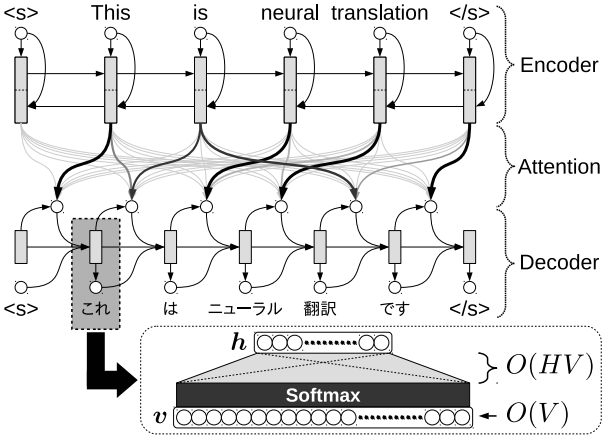


図 1: NMT モデルの例, 及び出力層の計算量

には V として数万程度が設定されるため [1], 出力層で記憶領域を多く消費してしまう問題がある.

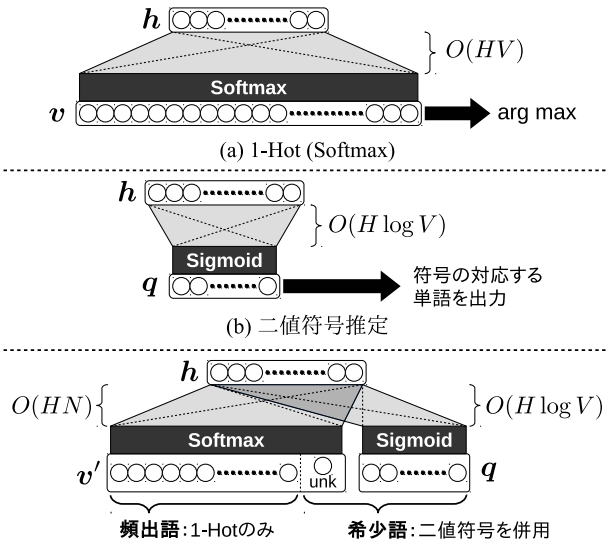
これに対し提案法では, 1-Hot 表現のように単語ごとに個別の値を推定するのではなく, 各単語に特定のビット列を割り当て, 出力層がこのビット列を予測することで間接的に単語の推定を行う. まず, 単語 w に割り当てるビット列を $\boldsymbol{b}(w) := [b_1(w), b_2(w), \cdots, b_B(w)]$ と表記する. ここで各 $b_i(w)$ は w ごとに 0 か 1 を返す二値関数であり, B は符号全体のビット数である. V 種類の単語の識別には $\boldsymbol{b}(w)$ が少なくとも V 種類の異なる元を持つ必要があり, ここから $B \geq \lceil \log_2 V \rceil$ の条件が得られる. 出力層では隠れ層の値 $\boldsymbol{h}$ から, 各 $b_i(w)$ に対応する B 個の確率値 $\boldsymbol{q}(\boldsymbol{h}) := [q_1(\boldsymbol{h}), q_2(\boldsymbol{h}), \cdots, q_B(\boldsymbol{h})]$ を複数の Logistic 回帰

$$\boldsymbol{q}(\boldsymbol{h}) := \sigma(W_{hq}\boldsymbol{h} + \boldsymbol{\beta}_q), \quad \sigma(\boldsymbol{x}) := 1/(1 + \exp(-\boldsymbol{x}))$$

により独立に求める. ここで $W_{hq} \in \mathbb{R}^{B \times H}, \boldsymbol{\beta}_q \in \mathbb{R}^B$ は学習可能なパラメータである. q_i を i 番目のビットが 1 になる確率と見なせば, 各単語 w の生成確率 $\Pr(w|\boldsymbol{h})$ はその総乗

$$\Pr(w|\boldsymbol{h}) := \prod_{i=1}^{B} [b_i q_i + (1 - b_i)(1 - q_i)]$$

として得られる. 以降, 式中の引数は適宜省略する. $\boldsymbol{q}$ から確率最大となるビット列を得るには, 単に $q_i \geq 1/2$ であれば $b_i = 1$, そうでなければ 0 とすればよいが, 単語の割り当てに使用する符号化手法によっては単語と対応しないビット列が推定されてしまう可能性がある



(a) 1-Hot (Softmax)

(b) 二値符号推定

(c) 1-Hot + 二値符号推定

図 2: 二値符号予測を用いた出力層の構成

点に注意する. 本研究ではそのような出力が得られた場合, 単に未知語記号に変換するものとした.

$\boldsymbol{b}$ としては多様な符号化手法の適用が考えられるため, 事前実験として簡単なコーパスを用いて数種類の手法による翻訳精度の調査を行った [2]. この結果, 単語のコーパス中の出現頻度の順位を二進数化したものを使用したとき比較的高い翻訳精度となったため, 本論文ではこの手法を使用する. この手法により生成されるビット列は $B = \lceil \log_2 V \rceil$ を達成し, 最小の符号化手法の一つであると共に, 上位ビットによって単語のおおよその出現頻度が定まるという特徴がある. 例えば $V = 16 = 2^4$ かつ出現頻度が 0 から数えて 5 番目の単語の場合, 対応するビット列は $\boldsymbol{b} = [0101]$ となる.

$\boldsymbol{q}$ に関する損失関数としては, $\boldsymbol{b}$ との二乗和誤差

$$L_{\mathcal{B}}(\boldsymbol{q}, \boldsymbol{b}) := \sum_{i=1}^{B} (q_i - b_i)^2$$

または交差エントロピー

$$L_{\mathcal{B}}(\boldsymbol{q}, \boldsymbol{b}) := -\sum_{i=1}^{B} [b_i \log q_i + (1 - b_i) \log(1 - q_i)]$$

等が使用可能であり, これら損失関数の違いに関しても事前実験による検討を行った. この結果, いずれの手法についても二乗和誤差を用いた場合に比較的高い翻訳精度となったため, 本論文では $\boldsymbol{q}$ に関する損失関数を二乗和誤差に統一した.

図 2(a) に従来の 1-Hot 表現による出力層, 図 2(b) に提案法の概略を示す. 提案法が出力層に用いるパラメータ $W_{hq}, \boldsymbol{\beta}_q$ の空間計算量は B が最小の符号化手法で $O(H \log V)$ となり, 従来法の $O(HV)$ と比較して非常に小さく抑えられる. 例えば V が数万程度であれば, 実際のパラメータ数は従来法の数千分の 1 となる.

なお, Hierarchical Softmax [3] において同一階層内のノードでパラメータを共有し, 階層間の依存関係を無視した場合, 提案法の $\boldsymbol{b}$ として二分木に基づく符号を選択した場合と一致する. このため, 提案法は Hierarchical Softmax にビット間の独立性の制約を与えた特殊形としても解釈可能である. この制約によって全ビットが同時に計算可能であるため, GPU などの並列計算に特化した環境に適するという利点がある.

しかし実験で示すように, 純粋に単語とビット列を個別に対応させただけの符号を出力層で学習した場合, 1-Hot 表現ほど適切なモデルを獲得できず, 翻訳精度が大幅に低下してしまう問題がある. 2.2, 2.3 節では, 本節の基本的な手法に付加的な処理を行うことで, 出力層の予測精度を向上させる手法を述べる.

2.2 1-Hot 表現と二値符号の混合モデル

コーパス中に含まれる単語は出現頻度が大きく偏っている [4]. このため提案法をそのまま適用した場合, 頻出語に関するビット列を多く学習することとなり, 希少語に必要な表現をネットワークがうまく獲得できない可能性がある. この問題に対処するため, 本節では 1-Hot 表現と二値符号予測を組み合わせたモデルを導入する. 具体的には図 2(c) で示すように, まず出現頻度が上位 $N-1$ 番目までの頻出語と「その他」について 1-Hot 表現による推定を行い, 「その他」と推定されたときのみ二値符号による推定を行う. $\mathrm{id}(w)$ が出現頻度順に対応するとして, 各単語の生成確率は

$$\begin{aligned}
\Pr(w|\boldsymbol{h}) &:= \begin{cases} v'_{\mathrm{id}(w)} & \text{if } \mathrm{id}(w) < N \\ v'_N \cdot \pi(w, \boldsymbol{h}) & \text{otherwise} \end{cases} \\
\boldsymbol{v}' &:= \exp \boldsymbol{u}' / \operatorname{sum} \exp \boldsymbol{u}' \\
\boldsymbol{u}' &:= W_{hu'} \boldsymbol{h} + \boldsymbol{\beta}_{u'} \\
\pi(w, \boldsymbol{h}) &:= \prod_{i=1}^{B} [b_i q_i + (1 - b_i)(1 - q_i)]
\end{aligned}$$

で定義される. ここで $W_{hu'} \in \mathbb{R}^{N \times H}, \boldsymbol{\beta}_{u'} \in \mathbb{R}^N$ は学習可能なパラメータである. また出力層の損失関数は

$$L := \begin{cases} \lambda_{\mathcal{H}} L_{\mathcal{H}}(\boldsymbol{v}', \mathrm{id}(w)) & \text{if } \mathrm{id}(w) < N \\ \lambda_{\mathcal{H}} L_{\mathcal{H}}(\boldsymbol{v}', N) + \lambda_{\mathcal{B}} L_{\mathcal{B}}(\boldsymbol{q}, \boldsymbol{b}) & \text{otherwise} \end{cases}$$

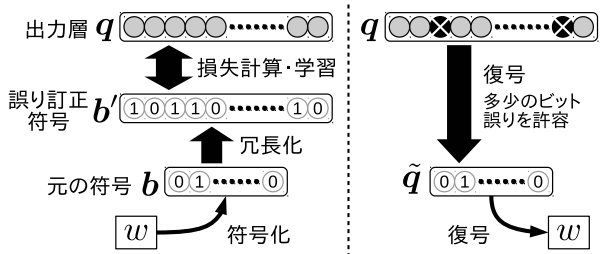
とする. ここで $\lambda_{\mathcal{H}}, \lambda_{\mathcal{B}}$ は 1-Hot 部と二値符号部の学習重みを決める係数だが, 本研究では簡単のため $\lambda_{\mathcal{H}} = \lambda_{\mathcal{B}} = 1$ とした.

この混合モデルの計算量は $O(H(N + \log V))$ であり, 1-Hot 部の影響で二値符号予測のみの場合より計算量が増大する. しかし 1-Hot 部が扱うのは頻出語のみであるため, N としては数百程度の小さな値を指定可能である. このため, 1-Hot 表現のみの出力層と比較した場合には計算量が大きく削減されることとなる.

なお, Chen らは単語を出現頻度ごとにグループ化し, 各グループに対して隠れ層の異なる部分を使用して確率の推定を行うことでパラメータ数を削減する手法を提案している [5]. 頻出語と希少語を別々に扱う考えは Chen らの手法に基づくが, 提案法では隠れ層の全ての要素を出力層の計算に使用する点が異なる. これは提案法の場合, 混合モデルを適用しても出力層全体の計算量が十分小さく抑えられるため, 出力層を全結合ネットワークとしても問題にならないためである.

2.3 誤り訂正符号による冗長化

前節までの手法は, 推定したい単語によって $\boldsymbol{q}$ の全要素が値に制約を受けるため, 僅かな推定誤りが原因で全く異なる単語を生成してしまう可能性がある. そ



(a) 学習時の動作　(b) 生成時の動作

図 3: 誤り訂正符号による出力層の冗長化

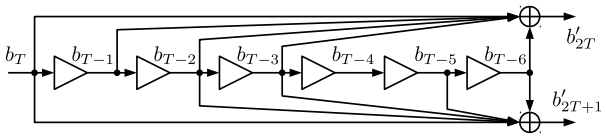


図 4: 実験で使用した畳み込み符号

こで, 同じ単語に対して Hamming 距離の近い複数のビット列を割り当てることで, 多少のビット誤りに頑健な単語の推定が可能になると考えられる. Dietterich らは学習対象を誤り訂正符号とする手法が有効であると報告しており [6], これに基づき, 本研究では誤り訂正符号を用いた出力ビット列の冗長化を導入する.

具体的には図 3 で示すように, 元のビット列 $\boldsymbol{b}$ に対して適当な誤り訂正符号による冗長化を適用し, その結果生成される新たなビット列 $\boldsymbol{b}' := [b'_1(\boldsymbol{b}), b'_2(\boldsymbol{b}), \cdots, b'_{B'(B)}(\boldsymbol{b})]$ を出力層で学習する. ここで $B'(B) \geq B$ は誤り訂正符号の出力ビット数である. 典型的な誤り訂正符号は $O(B'/B) = O(1)$ であり, 冗長化の有無による出力層の計算量の変化は高々定数倍に抑えられる. 翻訳結果を実際に生成する際は $\boldsymbol{q}$ に対して復号処理を行い, 得られた結果 $\tilde{\boldsymbol{q}} := [\tilde{q}_1(\boldsymbol{q}), \tilde{q}_2(\boldsymbol{q}), \cdots, \tilde{q}_B(\boldsymbol{q})]$ を元のビット列の確率として解釈することで単語の復元を行う.

ここで, 適用する誤り訂正符号としては畳み込み符号 [7] が適切であると考えられる. これは畳み込み符号がランダムなビット誤りに対して特に頑健に動作する点と, 確率過程に基づく復号手法が存在し, 出力層が生成する実数値を直接扱えるという特徴による. また畳み込み符号の典型的な手法は入力ビット数に依存しない形であるため, $\boldsymbol{b}$ の符号化手法が誤り訂正符号側から制約を受けにくいという利点もある.

畳み込み符号は一度に考慮するビット数に応じて任意の冗長性を得られるが, 同時に復号処理の計算量が増大する. このため, 誤り訂正能力と計算量のトレードオフを考慮して手法を設計する必要がある. 本研究では事前に数種類の畳み込み符号を検討し, 時間計算量が無視できる範囲で最も復号能力の高かった図 4 の手法を採用した. この符号は入力ビット列に 2 種類の窓関数を畳み込み, B ビットから $2(B+6)$ ビットの符号を生成する. 復号には複数の手法が存在するが, 本研究では実装の容易な硬判定 Viterbi 法を使用した.

3 実験

3.1 実験設定

提案法による翻訳精度を評価するため, 難易度の異なる 2 種類の翻訳タスク (ASPEC [8] 及び BTEC [9]) において BLEU [10] の値を評価した. 表 1 に各コーパスの詳細を示す. コーパスのトークン化は英語に tokenizer.perl[1], 日本語に KyTea [11] を使用した.

表 1: 実験に使用したコーパスの詳細

コーパス		ASPEC	BTEC
言語対		En → Ja	
文数	Train	2.00 M	465 k
	Dev	1,790	510
	Test	1,812	508
語彙数（両言語）		65536	25000

NMT のフレームワークとして NMTKit[2] を使用し, NVIDIA GeForce GTX TITAN X を各 1 台使用して学習を行った. 提案法の変更点は出力層のみであり, 翻訳モデルの他の部分に関しては Luong らによる Concat モデル [1] を使用した. 出力層以外の隠れ層は全て 512 次元とし, RNN の層数は 1 とした. また RNN の入出力層のみ 30%の Dropout [12] を行った. 最適化には Adam [13] の推奨設定を使用し, 学習率は固定した. コーパスは事前に両言語の単語数でソートを行い, 64 文ごとにミニバッチを作成した. BLEU の評価はミニバッチ 1000 個を学習するごとに行った.

実験を行った出力層の構成法は以下の通りである.

1-Hot ··· 1-Hot 表現 (=Softmax, 従来法)
Binary ··· 二値符号
Hybrid ··· 1-Hot 表現 ($N = 512$) と二値符号の混合
Binary+ECC ··· Binary を誤り訂正
Hybrid+ECC ··· Hybrid の二値符号部を誤り訂正

3.2 実験結果

表 2 に各手法による Test データ上での BLEU, 出力層で推定する二値符号の符号長 B, 出力層のパラメータ数 $\#_{W,\boldsymbol{\beta}}$ とその 1-Hot との比率, 及びモデル全体が使用するパラメータ数の 1-Hot との比率を示す. また図 5, 6 に各コーパスにおける Test データ上での BLEU の学習中の変化を示す. 表 2 の BLEU に関しては値の局所的な不安定さを緩和するため, Dev データ上で最大値となった世代の前後 2 世代を含む計 5 世代についての Test データ上での平均値を用いた.

まず, いずれの提案法についても従来法と比較して出力層のパラメータ数が数十から数千分の 1 程度まで大きく減少していることが分かる. モデル全体で見ると, 提案法では約 30%程度[3] のパラメータが削減されており, 実質的には従来法で出力層のために確保されていた分を無視した程度のパラメータ数であると考えることができる. 特に V の大きな ASPEC では従来法と提案法の差が顕著であり, 提案法で最も巨大なパラメータ数を持つ Hybrid+ECC でも, 出力層の大きさが従来法の 1/100 未満に抑えられている.

BLEU に関しては前章で議論したように, 単純に二値符号予測のみを使用した場合 (Binary) は他と比較して顕著に低い値となっている. 一方, 混合モデル

[1] http://www.statmt.org/moses/?n=moses.baseline
[2] https://github.com/odashi/nmtkit
[3] 残りのパラメータの大部分は Encoder 及び Decoder の入力層における単語ベクトルであり, 翻訳時には単語と無関係な大部分の値へのアクセスを省略可能である. このため, これらのパラメータは必ずしもメモリ上に常に保持する必要がない点に注意する.

表 2: 各手法による BLEU, 符号長 B, 出力層のパラメータ数 $\#_{W,\beta}$ と 1-Hot との比率

手法	ASPEC					BTEC				
	BLEU%	B	$\#_{W,\beta}$	1-Hot との比率 (出力層)	1-Hot との比率 (全体)	BLEU%	B	$\#_{W,\beta}$	1-Hot との比率 (出力層)	1-Hot との比率 (全体)
1-Hot	30.75	—	33.6 M	1/1	1	47.72	—	12.8 M	1/1	1
Binary	13.78	16	8.21 k	1/4.10 k	0.698	31.83	15	7.70 k	1/1.67 k	0.738
Hybrid	22.81	16	271. k	1/124.	0.700	44.23	15	270. k	1/47.4	0.743
Binary+ECC	25.95	44	22.6 k	1/1.49 k	0.698	44.48	42	21.5 k	1/595.	0.738
Hybrid+ECC	29.07	44	285. k	1/118.	0.700	47.20	42	284. k	1/45.1	0.744

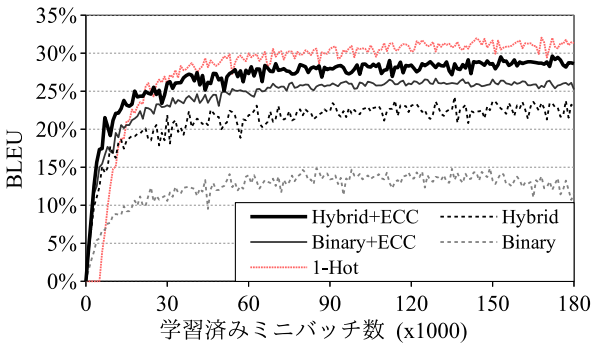


図 5: 学習時の BLEU の変化 (ASPEC)

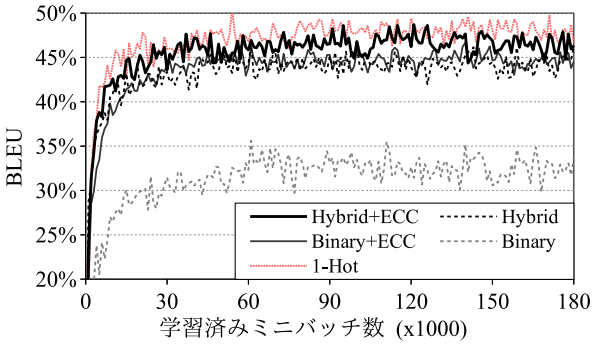


図 6: 学習時の BLEU の変化 (BTEC)

(Hybrid) や誤り訂正符号の適用 (Binary+ECC) で翻訳精度の大幅な改善が見られ, それぞれの手法が二値符号予測に対して効果的に作用することが確認できる. Hybrid と Binary+ECC を比較した場合, BTEC では同程度, ASPEC では Binary+ECC の方が高い BLEU を記録している. Binary+ECC のパラメータ数は Hybrid の 1/10 未満であり, このことから誤り訂正符号の適用により 1-Hot よりも少ないパラメータ数で効率的に翻訳精度を改善可能であることが分かる.

また混合モデルと誤り訂正符号を併用した場合 (Hybrid+ECC) では更に改善が見られ, BTEC の場合, 1-Hot の 1/45 程度のパラメータ数でほぼ同じ BLEU を達成している. より難易度の高い ASPEC に関しては 1-Hot から 2, 3 ポイント低い値を記録しているが, パラメータ数が 1/100 未満に削減されていることを考慮すれば, 十分に高い性能を持つモデルであると言える.

4 おわりに

本研究では, NMT の出力層に二値符号を用いた表現を採用することで, 従来法で問題となっていた出力層の空間計算量を大幅に削減する手法を提案した. 実験により, 提案法による出力層が従来法と比較してパラメータ数を数十から百分の 1 未満に抑えながら, 従来法に匹敵する翻訳精度を達成可能である点を示した.

提案法は任意の二値符号に対して適用可能であるが, より翻訳に適した符号を選択すれば翻訳精度を向上可能であると考えられる. また削減された計算量をより高度なネットワークの構築に転用する等も有効であると考えられる. これらの応用的な手法, また提案法の最適な設定に関する詳しい調査は今後の課題である.

また提案法の出力は確率に基づいて探索を行うことで k-best 解が容易に得られるため, 従来法と同様にビーム探索や翻訳候補に基づく最適化 [14] を実行可能である. これらの影響についても今後調査を行いたい.

謝辞

本研究の一部は JSPS 科研費 15J10649, 及び 16H05873 の助成を受けたものである.

参考文献

[1] T. Luong, H. Pham, and C. D. Manning. Effective approaches to attention-based neural machine translation. In *Proc. EMNLP*, 2015.

[2] 小田悠介, P. Arthur, G. Neubig, 中村哲. 二値符号予測によるニューラルネット翻訳. NLP 若手の会第 11 回シンポジウム, 2016.

[3] F. Morin and Y. Bengio. Hierarchical probabilistic neural network language model. In *Proc. AISTATS*, 2005.

[4] G. K. Zipf. Human behavior and the principle of least effort. 1949.

[5] W. Chen, D. Grangier, and M. Auli. Strategies for training large vocabulary neural language models. In *Proc. ACL*, 2016.

[6] T. G. Dietterich and G. Bakiri. Solving multiclass learning problems via error-correcting output codes. *J. Artif. Int. Res.*, Vol. 2, No. 1, pp. 263–286, 1995.

[7] A. Viterbi. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE Trans. Inf. Theor.*, Vol. 13, No. 2, pp. 260–269, 1967.

[8] T. Nakazawa, M. Yaguchi, K. Uchimoto, M. Utiyama, E. Sumita, S. Kurohashi, and H. Isahara. ASPEC: Asian scientific paper excerpt corpus. In *Proc. LREC*, 2016.

[9] T. Takezawa. Building a bilingual travel conversation database for speech translation research. In *Proc. Oriental COCOSDA*, 1999.

[10] K. Papineni, S. Roukos, T. Ward, and W. Zhu. BLEU: A method for automatic evaluation of machine translation. In *Proc. ACL*, 2002.

[11] G. Neubig, Y. Nakata, and S. Mori. Pointwise prediction for robust, adaptable japanese morphological analysis. In *Proc. ACL-HLT*, 2011.

[12] N. Srivastava, G. E Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, Vol. 15, No. 1, pp. 1929–1958, 2014.

[13] D. Kingma and J. Ba. Adam: A method for stochastic optimization. In *Proc. ICLR*, 2015.

[14] S. Shen, Y. Cheng, Z. He, W. He, H. Wu, M. Sun, and Y. Liu. Minimum risk training for neural machine translation. In *Proc. ACL*, 2016.