二値符号予測と誤り訂正を用いたニューラル翻訳モデル

小田 悠介[†,††]・Philip Arthur[†]・Graham Neubig[†††,†]・吉野 幸一郎[†,††††]・中村 哲[†]

本論文では，ニューラル翻訳モデルで問題となる出力層の時間・空間計算量を，二値符号を用いた予測法により大幅に削減する手法を提案する．提案手法では従来のソフトマックスのように各単語のスコアを直接求めるのではなく，各単語に対応付けられたビット列を予測することにより，間接的に出力単語の確率を求める．これにより，最も効率的な場合で従来法の対数程度まで出力層の計算量を削減可能である．このようなモデルはソフトマックスよりも推定が難しく，単体で適用した場合には翻訳精度の低下を招く．このため，本研究では提案手法の性能を補償するために，従来法との混合モデル，および二値符号に対する誤り訂正手法の適用という 2 点の改良も提案する．日英・英日翻訳タスクを用いた評価実験により，提案法が従来法と比較して同等程度の BLEU を達成可能であるとともに，出力層に要するメモリを数十分の 1 に削減し，CPU での実行速度を 5 倍から 10 倍程度に向上可能であることを示す．

キーワード：ニューラル翻訳，二値符号予測，誤り訂正

Neural Machine Translation Models using Binarized Prediction and Error Correction

Yusuke Oda[†,††], Philip Arthur[†], Graham Neubig[†††,†], Koichiro Yoshino[†,††††] and Satoshi Nakamura[†]

In this paper, we propose a new method for calculating the output layer in neural machine translation systems with largely reduced computation cost based on binary code. The method is performed by predicting a bit array instead of actual output symbols to obtain word probabilities, and can reduce computation time/memory requirements of the output layer to be logarithmic in vocabulary size in the best case. In addition, since learning proposed model is more difficult than softmax models, we also introduce two approaches to improve translation quality of the proposed model: combining softmax and our models and using error-correcting codes. Experiments on English-Japanese bidirectional translation tasks show proposed models achieve that their BLEU approach the softmax, while reducing memory usage on the order of one tenths, and also improving decoding speed on CPUs by x5 to x10.

Key Words: *Neural Machine Translation, Binalized Prediction, Error Correction*

† 奈良先端科学技術大学院大学, Nara Institute of Science and Technology
†† 情報通信研究機構, National Institute of Information and Communications Technology
††† カーネギーメロン大学, Carnegie Mellon University
†††† 科学技術振興機構, Japan Science and Technology Agency

自然言語処理 Vol. 13 No. 2 **April 2006**

1 はじめに

機械翻訳システムでより多くの文を対象に翻訳精度を維持したい場合，その量に応じた大きさの語彙をシステムが取り扱う必要がある．語彙サイズは様々な機械翻訳手法の性能や効率に影響を及ぼすが，特に近年活発に研究されているニューラル翻訳モデル (Sutskever, Vinyals, and Le 2014) では，語彙サイズの増加に伴う影響が顕著である．図 1 はエンコーダ （Encoder: 符号化器）, デコーダ （Decoder: 復号器）および注意機構（Attention）と呼ばれる個々のネットワーク構造からなる翻訳モデル (Bahdanau, Cho, and Bengio 2014; Luong, Pham, and Manning 2015) であり，ニューラル翻訳モデルとして典型的に使用される構造である．エンコーダは入力シンボル列を連続空間上のベクトル集合に変換し，この情報をもとにデコーダが出力シンボルを 1 個ずつ順に決定する．エンコーダとデコーダの内部構造はモデルによって様々であり，典型的には複数のリカレントニューラルネットワーク（Recurrent Neural Network: RNN）を用いて構成される．注意機構はエンコーダが生成したベクトルに関する重み付き和を与えるモデルで，デコーダが次回のシンボル推定に使用する文脈情報を生成する．

ここで，ニューラルネットワークで単語等の離散的なシンボルを扱う場合，モデルの入出力層でシンボルと内部ベクトルとの相互変換を行う必要がある．この特徴は特に出力層側で問題となる．入力層側は毎回特定の単語が与えられるため，無関係な単語に関する計算は行われないのに対し，出力層側はあらゆる候補の中から妥当な出力単語を選択する必要があるためである．単語選択のアルゴリズムとして語彙サイズに対する時間・空間計算量の大きな手法を選択した場合，実質的な計算コストが語彙サイズに依存することとなり，翻訳モデルを構築・運用する上での問題となる．実際，ニューラルネットワークによる単語推定で最も単純かつ標準的な手法であるソフトマックス演算は，語彙に含まれる全単語のスコアを隠れ層の一次結合として愚直に計算するため，計算量は語彙サイズに比例する．このため，出力層の計算をいかにして軽量化するかが重要な課題であると言える．

この問題はよく認識されており，2.2 で紹介するように，従来様々な解決手法が提案されてきた．出力層を改良するにあたっての着眼点は様々であり，従来手法が何を重点的に解決しようとしているかはそれぞれ異なる．この中で，特に重要と考えられる 4 つの観点を以下に示す．

翻訳精度 手法を適用した際，平均的な翻訳精度が大幅に低下してはならない．特に，単純なソフトマックスと比較して同等程度の性能が維持可能，あるいは，可能であればより高い性能を達成可能である手法が望ましい．

空間効率 (使用メモリ量) 膨大なメモリを必要とする手法を実行するためには大規模かつシステムが専有可能な計算資源が必要であり，携帯デバイス等の計算資源の制約の強い機器での直接実行には適さない．多くの環境に搭載可能なシステムを構築するためには，手法自体が可能な限り少ないメモリ消費の下で動作可能である必要がある．

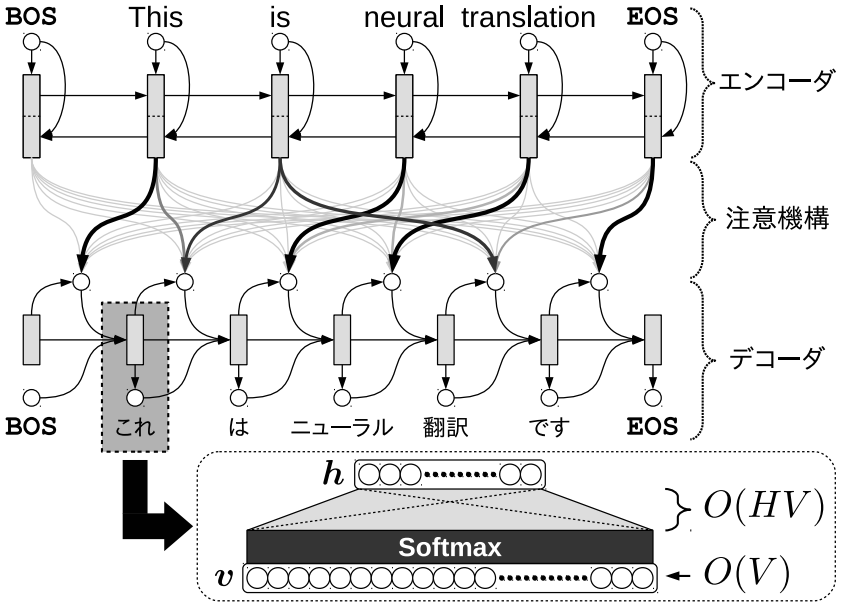

図 1　エンコーダ・デコーダモデルに注意機構を導入した典型的なニューラル翻訳モデルの概観．このうち，出力層における内部ベクトルから単語への変換が大きな計算負荷となる．

時間効率 (実行速度)　可能な限り高速に動作する手法が望ましい．高速にパラメータを学習可能であればシステムをチューニングする利便性が向上し，また運用時に高速なシステムは計算資源やユーザへの負担を減少させることとなる．空間効率と同様に，運用時に強力な計算資源が使用可能とは限らず，このため非力な CPU でも効率的に動作可能な手法がより望ましい．

並列計算との親和性　運用時とは対照的に，パラメータの学習時には GPU 等の強い並列性を持つ計算資源を使用することができる場合がある．並列化の容易な手法であれば，学習時にこれらの強力な計算資源の恩恵に与ることが可能である．

これらの観点のうち，いずれの項目を特に重視するかが手法自体の特徴となる．提案手法では特に空間効率と時間効率に関して，モデルの定式化段階での計算量を削減することに主眼を置き，翻訳精度は既存手法で最も表現力の高いソフトマックスモデルと同等程度の実現を目標とした．

提案手法による出力層はソフトマックスとは異なり，語彙中の単語に対して直接スコアを計算することは行わない．その代わり，各単語に一意な二値符号を割り当て，そのビット列を単語の表現として出力層で学習することで，間接的に単語の推定を行う．この手法を用いることで，最も理想的な場合で 2^n 種類の単語を n ビットのみを使用して表現することが可能となる

ため，その推定に必要な時間・空間計算量を語彙サイズ V に対して $O(\log V)$ まで減少させることが可能となる．

提案手法の基本的なアイデアはこのように単純だが，実験で示すように，単に二値符号のみを用いる手法では翻訳精度が従来手法と比べて大幅に低下してしまうという問題がある．本論文では更に，この問題に対して 2 種類の観点から提案手法を改良する手法を導入する．まず，従来のソフトマックスモデルを部分的に導入することで，高頻度語と低頻度語を分離して学習可能とする手法を提案する．また，二値符号そのものの頑健性を向上させるために，誤り訂正符号，特に畳込み符号 (Viterbi 1967) による冗長化を施す．

実験では，二値符号予測とこれらの改善手法について，難易度の異なる 2 種類の英日・日英翻訳タスクを用いて翻訳精度の比較を行った．この結果より，提案手法が従来のソフトマックスと遜色ない翻訳精度を達成可能であるとともに，出力層の動作に必要なパラメータ数，および計算時間の両面においてソフトマックスよりも優れていることを示す．

2 詳細な問題定義と従来研究

2.1 単純なソフトマックスモデルの定式化

近年のニューラル翻訳モデルでは，多くのモデルがワンホット表現と呼ばれる単語の表示手法を入出力層に導入している．つまり，語彙サイズ V と同じ数の次元の連続空間 $\mathbb{R}^V$ を考え，このうちある単語 ID $\mathrm{id}(w) \in \{x \in \mathbb{N} \mid 1 \leq x \leq V\}$ に対応する次元のみ 1，他の次元を 0 とする単位ベクトル $\boldsymbol{e}_{\mathrm{id}(w)} \in \mathbb{R}^V$ を単語の表現と見なす．本研究では特に出力層について着目するため，以降は単語や語彙に関する用語は全て目的言語側のみを指して用いることとする．さて，$\mathbb{R}^V$ の部分空間

$$\mathbb{R}^V_{\mathrm{Cat}} := \left\{ \boldsymbol{x} \mid \boldsymbol{x} \in \mathbb{R} \wedge \forall i. 0 \leq x_i \leq 1 \wedge \sum_{i=1}^{V} x_i = 1 \right\} \tag{1}$$

は V 次元のカテゴリカル分布の空間を表し，各 $\boldsymbol{e}_{\mathrm{id}(w)}$ はちょうどその頂点に位置する．ここから，ワンホット表現で得られるベクトルは特定の語彙を定義域とする確率質量関数の一種と見なすことができる．ここで添字付き細字変数 x_i はベクトル $\boldsymbol{x}$ の i 番目の要素を表し，以降の式においても，他のベクトルに対して同様の記法を用いるものとする．

典型的なニューラル翻訳モデルの出力層では，出力された V 次元ソフトマックス確率分布 $\boldsymbol{v} \in \mathbb{R}^V_{\mathrm{Cat}}$ と，ある単語 w のワンホット表現 $\boldsymbol{e}_{\mathrm{id}(w)}$ との交差エントロピーを最小化するようパラメータを学習する．図 1 に示すように，出力層は典型的には 1 層の全結合ネットワークと見

なすことができ，損失関数の計算は以下の手順となる．

$$
\begin{aligned}
L_{\mathcal{H}}(\boldsymbol{v}, \mathrm{id}(w)) &:= \mathcal{H}\left(\boldsymbol{e}_{\mathrm{id}(w)}, \boldsymbol{v}\right), && (2)\\
&= -u_{\mathrm{id}(w)} + \log \sum_{i=1}^{V} \exp u_i, && (3)\\
\boldsymbol{v} &:= \frac{\exp \boldsymbol{u}}{\sum_{i=1}^{V} \exp u_i}, && (4)\\
\boldsymbol{u} &:= W_{hu}\boldsymbol{h} + \boldsymbol{\beta}_u. && (5)
\end{aligned}
$$

ここで，$W_{hu} \in \mathbb{R}^{V \times H}$ と $\boldsymbol{\beta}_u \in \mathbb{R}^V$ は勾配法で学習可能なパラメータであり，$\boldsymbol{h}$ は出力層の計算に用いる隠れ層のベクトル，H は $\boldsymbol{h}$ の次元数を表す．また $\exp \boldsymbol{u}$ はベクトル $\boldsymbol{u}$ に関する要素ごとの指数関数であり，他のベクトルに関しても同様の表記を用いるものとする．ワンホット表現はただ一つの要素が 1 であるベクトルのため，式 (2) を交差エントロピーの定義に従って分解することで式 (3) が得られる．複数の隠れ層から出力が計算されるネットワークの場合は，$\boldsymbol{h}$ として出力層に直結する全てのベクトルを結合したものを考えればよい．式 (5) より明らかに，単純なソフトマックスモデルの計算には $O(HV)$ だけの時間・空間計算量が必要となる．このうち，隠れ層の次元数 H は典型的には数百から千程度で決め打ちされることが多いのに対し，語彙サイズ V は使用するコーパスによって数千から数万程度の値を選択することとなり (Sutskever et al. 2014)，出力層の計算量に直接影響を与える．

2.2 出力層の軽量化に関する従来手法

出力層の軽量化は，ニューラル翻訳モデルと同様のネットワーク構造を持つモデル群の中心的な話題のひとつであり，様々な着眼点に基づく手法が提案されている．これらの着眼点は以下のようなグループに分類でき，グループの異なる手法はいくつかの例外を除き基本的に併用可能である．

後処理による隠れ層の大きさの削減　前述のように，ソフトマックスモデルの計算量は時間・空間ともに $O(HV)$ となる．本グループに属する手法は，学習済みのモデルに対して後処理を行い，隠れ層のサイズ H の大きさを縮小することで，全体的な計算量の圧縮を行うことを目的としている．また，出力層の直前のみでなく，モデルの隠れ層全体を削減する手法についても本グループに属するものと考えられる．

このような考えに基づく一般的な手法としては，学習済みパラメータのノルムを比較し，実際の計算への寄与が小さいものを削除してしまう**重み枝刈り（Weight Pruning）**(See, Luong, and Manning 2016)，学習済みモデルの最終的な出力を教師データとし，これを再現するより小さなモデルを再学習する**蒸留（Distillation）** (Kim and Rush 2016) などが挙げられる．いずれの手法においても，その着眼点は隠れ層側の圧縮であり，語彙

サイズ V に関しては元の計算量が維持される点に特徴がある．

確率分布の変形 本グループに属する手法は，隠れ層のサイズ H，及び語彙サイズ V を変えず，最終的な確率の定式化に変更を加えることで計算量を削減することを目的としており，後述するように本論文の提案手法はこのグループに属する．

階層的ソフトマックス（Hierarchical Softmax） (Morin and Bengio 2005) では，二分木を用いて単語を階層的にクラスタリングした後，木の根から左右どちらの枝を辿るかを二値分類問題として順に判定してゆくことで単語の推定を行う．このとき単語の生成確率は，木の根から単語に至るまでに経由した全ての分岐に関する二値分類確率の総乗として得られる．本手法を用いる場合，すべての単語に関する確率を求める操作は通常行わず，各分岐において選択されなかった側の枝に付属する単語は全て候補から除外される．このような貪欲探索を適用することで，選出される単語の大域最適性が保証されない代わりに，出力層の時間計算量を理想的には $O(H \log V)$ まで削減することが可能となる [1]．一方, 木に含まれる分岐の数は $V-1$ であり，それぞれが固有のパラメータを持つため，出力層全体の空間計算量はソフトマックスと同様 $O(HV)$ となり，パラメータの保持に必要なメモリの削減は期待できない．また木構造を使用することにより計算手順が複雑化するため，並列計算を行うには実装の工夫が必要となる．

階層的ソフトマックスは単語に対応するある種の二値符号を学習するモデルであると考えられるが，異なる観点から単語を二値符号化し，そのビット列を学習する手法として**ブルームフィルタ（Bloom Filter）**を適用する研究がある (Serrá and Karatzoglou 2017)．この手法ではフィルタ中の各ビットが 1 になる確率を尤度関数としてモデルの学習を行っているが，フィルタ自体の特性により，ビット列から単語を復元する写像の定義が曖昧であるため，機械翻訳をはじめとした文生成タスクへ直接適用するのは難しいと考えられる．

区分ソフトマックス（Differentiated Softmax） (Chen, Grangier, and Auli 2016a) では，単語を複数のグループに分類し，それぞれのグループを隠れ層の異なる部分と対応させることで，比較的小さな行列の組み合わせで出力層を表現し，計算時間・使用メモリ量の両面での向上を実現している．これはパラメータ行列 W_{hu} を部分的にゼロ行列に固定することと等しい．ところが時間・空間計算量の観点では $O(HV)$ から変化がなく，語彙サイズ V が増加した際の拡張性の面では問題が残る．また，異なるグループに属する単語間の関連性を出力層で十分扱うことができず，全結合型のネットワークを用いるソフトマックスと比較すると改善の余地が残されている．

[1] 語彙に含まれる全単語のスコアを計算する必要がある場合，木に含まれる全ての分岐に対して評価が必要であり，このときの償却計算量は $O(HV)$ となる．

適応ソフトマックス（Adaptive Softmax） (Grave, Joulin, Cissé, Grangier, and Jégou 2017) では，単語を出現頻度に基づいて複数のグループに分類し，まず高頻度語のグループにおいて通常のソフトマックスを実行する．このときグループ内の単語とは別に，各低頻度語のグループそのものに対応するラベルを追加しておき，ソフトマックスによってそのラベルが選出された場合には該当するグループで再びソフトマックス計算を行う．この手法では，パラメータ数が低頻度語グループの推定の分だけ通常のソフトマックスから若干増加しており，推定に使用するグループ数を G とすると，パラメータに関する空間計算量は $O(H(V+G))$ となる．一方で，上位のソフトマックスで選択されなかったグループに属するパラメータは階層的ソフトマックスと同様に無視されるため，実質的な出力層の計算の大部分を高頻度語グループのソフトマックスのみで終了することができ，時間計算量の短縮が見込める．ただし，選出される単語の大域最適性がモデルの設定によっては保証されない可能性があり，この点も階層的ソフトマックスと同様である．
LightRNN (Li, Qin, Yang, and Liu 2016) は各単語に 2 種類の異なるワンホット表現を割り当て，単語推定を 2 つのソフトマックスの積に分解することで出力層の時間・空間計算量を $O(H\sqrt{V})$ に削減する手法である．この手法では 2 つのソフトマックスを RNN の異なる時刻に割り当てるため，RNN の系列長は通常のソフトマックスのちょうど 2 倍となるが，全体として階層的ソフトマックスより効率的に動作することを報告している．また，前回の学習結果を用いてワンホット表現の割り当てを修正することで，推定精度をより向上させる手法についても提案している．

サンプリング　出力層が生成する確率分布を全体の計算なしに近似する手法として，適当な数の不正解ラベルを選択し，これらと正解ラベルとの関係を損失として学習する**サンプリング法**が適用されることがある (Gutmann and Hyvärinen 2010; Mnih and Teh 2012; Mikolov, Sutskever, Chen, Corrado, and Dean 2013)．これらの手法は学習時に語彙サイズ V の時間計算量への影響を取り除くことができ，学習速度を向上させることが可能である一方，テスト時には元のモデルと同じ時間計算量を必要とする．空間計算量，特にパラメータ数に関しては元のモデルと同一であり，元のモデル構造を維持したまま学習時の時間計算量を削減する手法であると言える．また，確率分布の定式化手法によっては，サンプリング法の適用に制約を受ける場合がある．

サンプリング法の効果的な応用として，単語間で分散表現を共有することでパラメータ数を削減する手法が提案されている (Chen, Mou, Xu, Li, and Jin 2016b)．この手法では，各単語の分散表現を予め選出した共通単語の分散表現の線形和として定義することで，ネットワーク内のパラメータ数に関する空間計算量を語彙サイズ V から共通単語数 V'（論文では $V' = 8\text{k}$ に固定）に削減している．純粋なモデルの目的関数はソフトマックスと同一であるため，このままでは線形和の操作分だけ計算量が増加してしまうが，サ

ンプリング法を適用することでこの欠点を回避している．

語彙の変更 以上の手法がすべて語彙が与えられた下での軽量化手法であるのに対し，語彙の定義自体を変更することで V そのものを小さくしてしまう手法が適用されることがある．最も単純には語彙として**文字**を使用する手法 (Ling, Trancoso, Dyer, and Black 2015) が挙げられるが，これはエンコーダやデコーダの生成する系列長が単語を用いた場合と比べて著しく増加してしまう欠点がある．より効率的な手法として，学習データに含まれる全ての文字列を被覆する部分文字列の集合を学習し，これを語彙として使用する**サブワード（Subword）法** (Wu, Schuster, Chen, Le, Norouzi, Macherey, Krikun, Cao, Gao, Macherey, Klingner, Shah, Johnson, Liu, Kaiser, Gouws, Kato, Kudo, Kazawa, Stevens, Kurian, Patil, Wang, Young, Smith, Riesa, Rudnick, Vinyals, Corrado, Hughes, and Dean 2016; Sennrich, Haddow, and Birch 2016; Chitnis and DeNero 2015; Mikolov, Sutskever, Deoras, Le, Kombrink, and Černocký 2012) が挙げられる．語彙サイズを適切に設定し，コーパスに対し最適化されたサブワード列は，単語列と比較して遜色ない程度の系列長を実現できることが知られているが，効果的なサブワード集合の学習に関する知識が別途必要となる [2]．

3 二値符号予測に基づく単語推定モデル

本研究で提案する手法は，いずれも前節の着眼点のうち，確率分布の変形に基づいて計算量を削減する手法のグループに属する．本節では提案手法の基本的な定式化，及び単純な手法からの性能の改善手法について導入する．

3.1 二値符号を用いた単語の表現手法

図 2(a) は従来のソフトマックスモデルによる単語推定モデルを表す．また図 2(b) は本研究で提案する二値符号を用いた単語推定モデルを表す．提案手法はソフトマックスモデルとは異なり，各単語の生成確率をモデルが直接求めることは行わず，二値符号の各ビットの生成確率から間接的に単語の生成確率の推定を行う．まず，

$$\boldsymbol{b}(w) := [b_1(w), b_2(w), \cdots, b_B(w)] : \mathcal{V} \to \{0,1\}^B \tag{6}$$

を各単語 w に直接対応するビット列とする．ここで各 $b_i(w) \in \{0,1\}$ はそれぞれ独立した w に関する二値決定関数であり，B はビット列に含まれるこれらの関数の個数とする．また $\mathcal{V}$ は語

[2] `https://github.com/google/sentencepiece` に，Google 社の提案するサブワードモデルである SentencePiece に関するいくつかの実験結果が紹介されており，既存の形態素解析器と組み合わせた場合等，より発展的な応用についても議論されている（2017 年 9 月 30 日閲覧・GitHub コミット番号 73）．

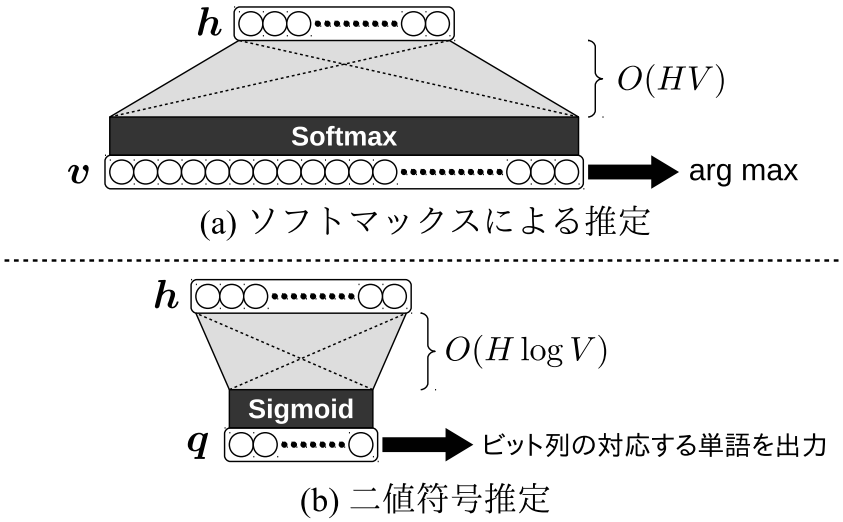

図 2　ソフトマックスモデルと提案手法における出力層の設計の相違点.

彙を表す．なお，本節における V は単語 ID $\mathrm{id}(w)$ の異なり数を表し，実際の語彙サイズ $|\mathcal{V}|$ とは異なることをここで注記しておく．

議論を明確にするために，$\boldsymbol{b}(w)$, およびビット列から単語への逆方向の写像 $b^{-1}(\cdot) : \{0,1\}^B \to \mathcal{V}$ に対して以下のような制約が存在するものとする．

- $\boldsymbol{b}(w)$ は単語 ID $\mathrm{id}(w)$ に関して単射である．つまり，

$$\mathrm{id}(w) \neq \mathrm{id}(w') \Rightarrow \boldsymbol{b}(w) \neq \boldsymbol{b}(w'). \tag{7}$$

  が成立する．この制約は，学習時に各単語を明確に区別して扱うために必要である．[3]
- $b^{-1}(\cdot)$ は左逆写像であり，定義域は $\{0,1\}^B$ 全体である．つまり $b^{-1}(\boldsymbol{b}(w)) = w$ が成り立つとともに，$\boldsymbol{b}(\cdot)$ の像に含まれるかどうかに関わらず，あらゆるビット列が何らかの単語に関連付けられているものとする．この制約は，翻訳モデルの不安定性により解釈不能なビット列が生成される可能性を排除するために必要であるとともに，4.2 節で導入する誤り訂正符号の根拠としても重要である．

これらの制約により，V 種類の単語を十分に識別するために必要なビット数として

$$B \geq \lceil \log_2 V \rceil \tag{8}$$

が自明な制約として得られる．

[3] 大文字・小文字等の表記の違いをどう扱うかは翻訳モデル自体の設計とは直接関係ないと考えられるため，これらの判断は $\mathrm{id}(w)$ の設計に隠蔽されているものとする．

提案手法の出力層では，$\boldsymbol{b}(w)$ の各ビットが 1 となる確率

$$\boldsymbol{q}(\boldsymbol{h}) := [q_1(\boldsymbol{h}), q_2(\boldsymbol{h}), \cdots, q_B(\boldsymbol{h})] \in [0,1]^B \tag{9}$$

を，式 (10) に示すように，現在の隠れ層の値 $\boldsymbol{h}$ からそれぞれ独立したロジスティック回帰モデルで推定する．

$$\boldsymbol{q}(\boldsymbol{h}) = \sigma(W_{hq}\boldsymbol{h} + \boldsymbol{\beta}_q), \tag{10}$$

$$\sigma(\boldsymbol{x}) := \frac{1}{1+\exp(-\boldsymbol{x})}. \tag{11}$$

ここで $W_{hq} \in \mathbb{R}^{B\times H}$ と $\boldsymbol{\beta}_q \in \mathbb{R}^B$ は学習可能なパラメータである．これより，ある単語 w が $\boldsymbol{q}(\boldsymbol{h})$ から生成される確率は，$\boldsymbol{q}(\boldsymbol{h})$ に含まれる各ビットごとの確率の積

$$\Pr(\mathrm{id}(w)|\boldsymbol{h}) := \prod_{i=1}^{B} (b_i(w)q_i(\boldsymbol{h}) + (1-b_i(w))(1-q_i(\boldsymbol{h}))) \tag{12}$$

として得られる．生成確率が最大となるビット列を $\boldsymbol{q}(\boldsymbol{h})$ から得るには，単に $q_i(\boldsymbol{h}) \geq 1/2$ である場合は 1, そうでなければ 0 を出力ビットとすればよい．

ここまでで記述したビット列に関する制約は非常に一般的なものであり，単語とビット列を対応させる手法には様々なものが考えられる．また，翻訳モデルにとってどのようなビット列の割り当て手法が効果的であるかは自明ではない．このため本研究では，事前実験として複数種の割り当て手法で実際に翻訳モデルの学習を行い，その中で経験的に最も高い翻訳精度を記録した Algorithm 1 に示す単語の出現頻度に基づく手法を採用した．[4] ここで，`UNK` (*unknown*) は語彙に含まれない単語，`BOS` (*begin-of-sentence*) は文頭記号，`EOS` (*end-of-sentence*) は文末記号である．また $\mathrm{rank}(w) \in \mathbb{N}_{>0}$ は学習データ中の出現頻度に基づく各単語の順位である．Algorithm 1 によるビット列の割り当ては最も効率的な手法 ($B = \lceil \log_2 V \rceil$) であることが保証される．また，高位のビットの組み合わせが単語のおおよその出現頻度を示していると考えられる一方，下位のビットほどランダム性が強まり，単語に関する頻度以外のどのような情報も保持していないと考えられる．

[4] Algorithm 1 以外に事前に実験した手法としては，完全なランダム，Huffman 符号 (Huffman 1952)， Brown Clustering (Brown, Desouza, Mercer, Pietra, and Lai 1992)，word2vec の単語ベクトルの符号に基づいた割り当て等がある．可変長の符号については末尾を 0 で埋めることで固定長として扱った．

Algorithm 1 実験で使用した単語からビット列への割り当て手法.

Require: $w \in \mathcal{V} \cup \{\texttt{UNK}, \texttt{BOS}, \texttt{EOS}\}$
Ensure: $\boldsymbol{b} \in \{0,1\}^B = w$ に対応するビット列

$$x := \begin{cases} 0, & \text{if } w = \texttt{UNK} \\ 1, & \text{if } w = \texttt{BOS} \\ 2, & \text{if } w = \texttt{EOS} \\ 2 + \mathrm{rank}(w), & \text{otherwise} \end{cases}$$

$b_i := \lfloor x / 2^{i-1} \rfloor \mod 2$
$\boldsymbol{b} \leftarrow [b_1, b_2, \cdots, b_B]$

3.2 損失関数

従来のモデルと同様に，提案手法による出力層は勾配法を用いて翻訳モデル全体と同時に最適化を行う．このため，ビット列のための損失関数はいたるところ (劣) 偏微分可能であり，また

$$L_{\mathcal{B}}(\boldsymbol{q}, \boldsymbol{b}) \begin{cases} = \epsilon_L, & \text{if } \boldsymbol{q} = \boldsymbol{b}, \\ \geq \epsilon_L, & \text{otherwise.} \end{cases} \tag{13}$$

を満たすべきである．ここで $\epsilon_L \in \mathbb{R}$ は損失関数の最小値であり，学習には影響しない．

明示的に確率モデルを学習する観点では，このような損失関数として交差エントロピー

$$L_{\mathcal{B}}(\boldsymbol{q}, \boldsymbol{b}) := -\sum_{i=1}^{B} \left(b_i \log q_i + (1 - b_i) \log(1 - q_i) \right), \tag{14}$$

を用いるのが望ましいが，事前実験により二乗誤差

$$L_{\mathcal{B}}(\boldsymbol{q}, \boldsymbol{b}) := \sum_{i=1}^{B} (q_i - b_i)^2, \tag{15}$$

を用いる方が最終的な翻訳精度が僅かに向上することが確認された．このため，実験では損失関数として式 (15) を使用することとした．

3.3 二値符号予測モデルの計算量

二値符号予測を用いた場合の出力層の計算量は，空間計算量・時間計算量ともにビット数 B に関して $O(HB)$ となる．ここで単語とビット列間の割り当てに Algorithm 1 で示したような最も効率的な手法を用いた場合，この計算量は $O(H \log V)$ に等しくなる．これは従来のソフトマックスモデルで必要とされた $O(HV)$ と比較して顕著に小さく，また V 種類のラベルを識別するモデルとしては最小の計算量であると考えられる．例として $V = 65536 = 2^{16}$ とした場合，出力層に必要とされるのは 16 ビット分のパラメータであり，従来法と比較して $16/65536 = 1/4096$ 程度までパラメータ数が削減されることとなる．

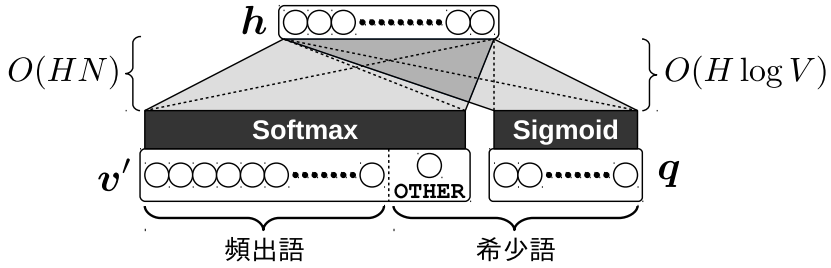

図 3 ソフトマックスと二値符号予測の混合モデル.

ここで, 二値符号予測モデルは従来手法である階層的ソフトマックス (Morin and Bengio 2005) に強い制約を導入したものと捉えることが可能である. 具体的には, 単語のビット列への割り当てを二分法の一種に基づいて行う点は階層的ソフトマックスと提案手法の共通点であり, 各ビットの推定にビット間の従属性を仮定しない点, および二分木上の同一階層で常に同じパラメータを使用する 2 点が異なる. これらの制約により, 提案手法のビット列 $\boldsymbol{b}$ の各ビットは完全に並列に計算することが可能であり, 式 (10) でも示したように, 実質的に単一の行列による積算として表現可能である. この特徴は, 提案手法が GPU などの並列計算に特化した計算資源上で特別な処理なしに計算可能であることを示している. また, ビット間の従属性を仮定しないため, ビットごとの独立した推定結果から常に大域最適なビット列を得られる点も階層的ソフトマックスと異なる.

4 二値符号予測モデルの改良

前節までで示した単純な二値符号予測には, 翻訳精度の点で問題が残っている. 実験で示すように, 二値符号予測を単体で用いた翻訳モデルは, 従来のソフトマックスモデルと比較して大幅に翻訳精度の低下を招くこととなる. 4.1 節と 4.2 節では, このような翻訳精度の問題を解決するために, 2 種類の改良を導入することで二値符号予測モデルの翻訳精度を向上させる手法を提案する.

4.1 ソフトマックスと二値符号予測の混合モデル

自然言語の単語のユニグラム出現頻度は Zipf の法則 (Zipf 1949) に従うことが知られており, 学習データ全体のほとんどが一部の頻出語のみに偏っていると言える. その結果, ニューラル翻訳モデルの出力層から伝えられる勾配も頻出語由来のものが多数となり, 語彙全体を効率よく学習できない可能性があるという問題がある. 従来のソフトマックスモデルでは, 各単語のスコアをそれぞれ独立したパラメータを用いて推定しており, 単語の出現頻度の偏りに起因す

る問題はこれらのパラメータ全体に分散する形で回避されていた．一方，提案手法である二値符号予測モデルでは同一のパラメータで頻出語と希少語の両方に対応する必要があり，学習機会の少ない希少語のビット列の学習を頻出語によって阻害されてしまう可能性がある．この問題を回避する単純な方法として，頻出語と希少語の予測をモデル的に分離し，一方の勾配が他方に影響を与えないようにすることが考えられる．このようなモデルとして，本節では図 3 に示すソフトマックスと二値符号予測の混合モデルを提案する．具体的には，図 3 左側のソフトマックス層で，上位 $N-1$ 位までの頻出語と「その他の単語」に相当する記号 OTHER を識別するモデルを学習する．希少語を出力するには，まず OTHER 記号をソフトマックス層で推定し，その後図 3 右側の二値符号推定により具体的な単語のビット列を推定する，という二段構造の推定を行う．ここで，ソフトマックス層と二値符号推定はそれぞれ独立したパラメータにより計算されるため，実際には並列計算が可能である点に注意する．この手法による各単語の生成確率は，ソフトマックス層の確率と二値符号推定による確率の積となり，

$$\Pr(w|\boldsymbol{h}) := \begin{cases} v'_{\mathrm{id}(w)}, & \text{if } \mathrm{id}(w) < N, \\ v'_N \cdot \pi(w, \boldsymbol{h}), & \text{otherwise}, \end{cases} \tag{16}$$

$$\boldsymbol{v}' := \frac{\exp \boldsymbol{u}'}{\sum_{i=1}^{V} \exp u'_i}, \tag{17}$$

$$\boldsymbol{u}' := W_{hu'}\boldsymbol{h} + \boldsymbol{\beta}_{u'}, \tag{18}$$

$$\pi(w, \boldsymbol{h}) := \prod_{i=1}^{B} \left(b_i q_i + (1-b_i)(1-q_i)\right), \tag{19}$$

と書くことができる．ここで $W_{hu'} \in \mathbb{R}^{N \times H}$ と $\boldsymbol{\beta}_{u'} \in \mathbb{R}^N$ は学習可能なパラメータであり，$\mathrm{id}(w)$ は単語の出現頻度の順位 $\mathrm{rank}(w)$ に基づいて定義されているものとする．また，損失関数はソフトマックス層の交差エントロピーと二値符号推定の損失の和で表記でき，

$$L := \begin{cases} l_{\mathcal{H}}(\mathrm{id}(w)), & \text{if } \mathrm{id}(w) < N, \\ l_{\mathcal{H}}(N) + l_{\mathcal{B}}, & \text{otherwise}, \end{cases} \tag{20}$$

$$l_{\mathcal{H}}(i) := \lambda_{\mathcal{H}} L_{\mathcal{H}}(\boldsymbol{v}', i), \tag{21}$$

$$l_{\mathcal{B}} := \lambda_{\mathcal{B}} L_{\mathcal{B}}(\boldsymbol{q}, \boldsymbol{b}), \tag{22}$$

となる．ここで $\lambda_{\mathcal{H}}$ と $\lambda_{\mathcal{B}}$ はソフトマックスと二値符号推定の学習重みを決めるハイパーパラメータである．これらは実際には調整が必要だが，本研究では簡単のために $\lambda_{\mathcal{H}} = \lambda_{\mathcal{B}} = 1$ のみを使用した．前述のように，希少語の推定にはソフトマックス層と二値符号予測の両者を使用するが，式 (20) の出力層における偏微分を考えれば，隠れ層 $\boldsymbol{h}$ より後段では両者の勾配が独立していることが分かる．

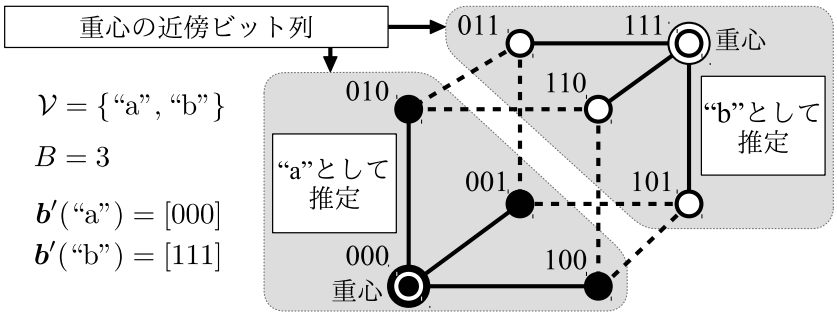

図 4 冗長な符号表現による予測の簡単な例.

混合モデルの計算量は，ソフトマックス層の追加により単純な二値符号予測モデルから増加し，$O(H(N + \log V))$ となる．ただしソフトマックス層は頻出語のみを対象とするため，N は V と比較して通常非常に小さく抑えられることとなり，実際の計算コストの増加はそれほど大きくならないことが期待される．ソフトマックス層の大きさを変化させたときの翻訳精度への影響については，実験で詳しく調査する．

混合モデルにより頻出語と希少語を分離するという考えは，区分ソフトマックス (Chen et al. 2016a) や適応ソフトマックス (Grave et al. 2017) で用いられた，単語クラスタごとに異なるスコア計算を用いる手法が元となっている．ただし，区分ソフトマックスでは隠れ層と出力層の結合に制約が設けられているのに対し，混合モデルでは全結合ネットワークを使用しており，隠れ層の値全てを各出力の推定に用いる点が異なる．これは，混合モデルを適用してもモデル全体の計算量を小さく抑えられるためである．また，混合モデルの希少語部をソフトマックスに置き換えた場合，適応ソフトマックスの分割数が 2 の場合とモデル的に一致する．このため，混合モデルは 2 分割適応ソフトマックスの計算量に改善を加えたものと捉えることも可能である．

4.2 誤り訂正符号の適用

前節までで述べた単純な二値符号予測モデル，および混合モデルは，使用する二値符号自体の頑健性を考慮していない点で問題がある．具体的には，$\boldsymbol{q}(\boldsymbol{h})$ は全てのビットを誤りなく推定しなければ正しい単語を予測することができず，1 ビット誤っただけで全く異なる単語を出力してしまう可能性がある．この問題は，ビット列全体の空間 $\{0, 1\}^B$ に全ての単語が密に配置されていることに起因するものであり，ビット列に対して何らかの冗長性を導入することで解決できるものと考えられる．

図 4 はこの考えを簡単な例で示したものである．ここでは 2 種類の単語 “a”, “b” を 3 ビットを使用して推定することを考え，各単語はそれぞれ「重心」ビット列 [000], [111] に写されるものと考える．このようなビット列の配置を選択すると，重心同士の間には 3 ビットのハミン

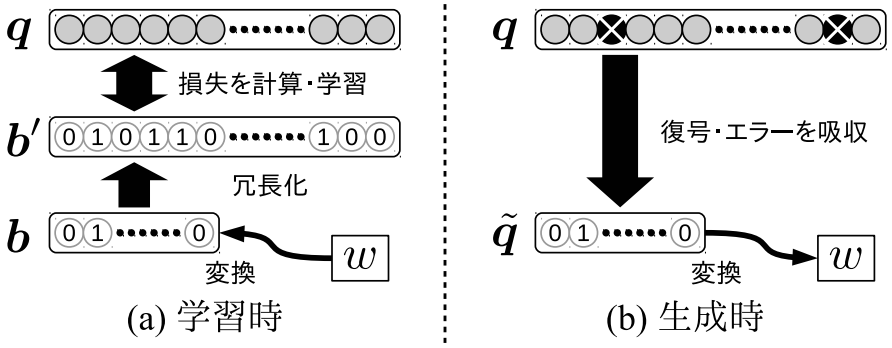

図 5　誤り訂正を導入した場合の学習時・テスト時の動作

グ距離を持つこととなり，1 ビットだけの間違いであれば，その近傍にある重心ビット列を選択することで元の単語を正しく復元可能であることが分かる．図 4 では各々の重心の近傍を灰色の領域で示している．このように，実際に記号が関連付けられたビット列同士の間隔に余裕を持たせることで多少のビット誤りを許容する手法は，誤り訂正符号 (Shannon 1948) の中心的な考えである．より具体的には，ビット列同士が少なくとも d ビットのハミング距離を持つとき，符号化手法全体では高々 $\lfloor (d-1)/2 \rfloor$ ビットまでの誤りを訂正可能であることが知られている．この距離は選択した誤り訂正手法によって定まる定数であり，最小ハミング距離または自由距離と呼ばれる．

誤り訂正をクラス分類問題に適用する考えは多く提案されており，タスクごとにうまく手法を設計することで，単純なクラス分類に対する優位性が示されている (Dietterich and Bakiri 1995; Klautau, Jevtić, and Orlitsky 2003; Liu 2006; Kouzani and Nasireding 2009; Kouzani 2010; Ferng and Lin 2011, 2013)．本研究では，特定の誤り訂正手法を Algorithm 1 で得られるビット列へ適用することで冗長化を行い，二値符号予測へ冗長性の導入を行う．ここで，過去に誤り訂正が検証されたクラス分類問題では高々 100 種類程度のクラスを扱っていたのに対し，本研究では語彙サイズに応じて数万種類程度のクラスを識別する必要があり，問題の複雑さが大きく異なる点に注意が必要である．本研究では，このような巨大なクラス分類問題においても誤り訂正手法の適用が有効であることを示す．

図 5(a) と 5(b) は誤り訂正手法を導入した際の学習時・テスト時の挙動を示している．学習時には，まず元となるビット列 $\boldsymbol{b}(w)$ を誤り訂正符号の重心となる冗長なビット列

$$\boldsymbol{b}'(\boldsymbol{b}(w)) := [\boldsymbol{b}'_1(\boldsymbol{b}(w)), \boldsymbol{b}'_2(\boldsymbol{b}(w)), \cdots, \boldsymbol{b}'_{B'}(\boldsymbol{b}(w))] : \{0,1\}^B \to \{0,1\}^{B'} \tag{23}$$

に写す．ここで $B'(B) \geq B$ は冗長化後のビット列に含まれるビット数である．この写像は単射であり，各 $\boldsymbol{b}(w)$ をより大きな空間のお互いに離れた位置に配置し直すこととなる．ニュー

ラル翻訳モデルはこの冗長なビット列 $\boldsymbol{b}'(\boldsymbol{b}(w))$ を出力層で学習する．なお，典型的な誤り訂正手法では B' は B の高々定数倍，つまり $O(B'/B) = O(1)$ であり，誤り訂正手法の適用前後で出力層の計算量自体が変化しない点は重要である．

翻訳モデルから実際の単語を生成する際は，モデルにより推定された確率列 $\boldsymbol{q}(\boldsymbol{h})$ を元に，元の単語に相当する確率列

$$\tilde{\boldsymbol{q}}(\boldsymbol{q}(\boldsymbol{h})) := [\tilde{\boldsymbol{q}}_1(\boldsymbol{q}(\boldsymbol{h})), \tilde{\boldsymbol{q}}_2(\boldsymbol{q}(\boldsymbol{h})), \cdots, \tilde{\boldsymbol{q}}_B(\boldsymbol{q}(\boldsymbol{h}))] : [0,1]^{B'} \rightarrow [0,1]^B \tag{24}$$

の復元を行う．このとき誤り訂正の効果により，$\boldsymbol{q}(\boldsymbol{h})$ に含まれる多少の誤りは許容され得ることとなる．写像 $\tilde{\boldsymbol{q}}(\cdot)$ は少なくとも $\boldsymbol{b}'(\cdot)$ の左逆写像であり，$[0,1]^{B'}$ 全体が定義域となる (復元結果が曖昧になるような入力が存在しない) ような手法を選択可能である．ここまでの議論で，3 節で導入したビット列に関する制約は誤り訂正の導入後も満たすことが分かる．このため，図 5 全体を単一の単語とビット列間の関連付け手法であると見なすことが可能である．

ここで，誤り訂正手法により冗長化されたビット列の特徴は，翻訳モデルの精度に直接影響を与えると考えられる．どのようなビット列が最適であるかは自明ではないが，ここでは望ましい特徴についての定性的な議論を行う．まず，図 4 は 1 ビット訂正可能な誤り訂正符号の一種であるが，ニューラルネットワークの出力としては明らかに適切でないことが直ちに分かる．なぜなら，重心ビット列の全てのビットが完全に一致しており，ニューラルネットワーク側から見ると各ビットの違いを区別できず，実質的に 1 ビットしか学習しない場合と同一のモデルになってしまうからである．このため，冗長化後のビット列は，ビット同士が可能な限りお互いに異なる傾向を持つことが望ましいと判断できる．また，出力層がどのようなビット誤りを生成するのかは自明ではないため，ランダムに発生する誤りを均等に訂正可能な手法を選択するのが妥当であると考えられる．更に，ニューラルネットワークが生成する出力 $\boldsymbol{q}(\boldsymbol{h})$ は 0 から 1 の間の連続値であり，単なるビット列よりも多くの情報を保持していると考えられる．このため，これらの連続値を直接使用してビット列を復元できる手法がより望ましい．

これらを満たす誤り訂正手法として，本研究では畳込み符号 (Viterbi 1967) を用いることとした．畳込み符号は入力ビット列とハイパーパラメータである重みビット列の畳込み和で表現され，ビット列中の離れた位置にあるビット同士が独立となるため，ランダムなビット誤りに特に頑健に動作するという特徴がある．また確率過程に基づくビット列の復号手法を用いることで，ビットごとの確率を直接考慮することが可能である．

Algorithm 2 は実験で実際に用いた畳み込み符号のアルゴリズムである．ここで，重みベクトル [1001111] および [1101101] は事前実験の結果により複数の設定から計算効率と復号能力のバランスを考慮して定めた．なお，$\boldsymbol{x}[i \,..\, j] := [x_i, \cdots, x_j]$，$\boldsymbol{x} \cdot \boldsymbol{y} := \sum_i x_i y_i$ である．畳込み符号は単一の冗長化手法に対して複数の復号手法が存在するが，実験では Algorithm 3 に示すアルゴリズムを使用した．ここで $\boldsymbol{x} \circ \boldsymbol{y}$ はベクトル $\boldsymbol{x}$ と $\boldsymbol{y}$ の結合を表す．Algorithm 3 はビタビア

Algorithm 2 実験で使用した畳み込み符号の冗長化アルゴリズム

Require: $\boldsymbol{b} \in \{0,1\}^B$
Ensure: $\boldsymbol{b}' \in \{0,1\}^{2(B+6)}$ = 冗長化されたビット列

$$x[t] := \begin{cases} b_t, & \text{if } 1 \le t \le B \\ 0, & \text{otherwise} \end{cases}$$
$$y_t^1 := x[t-6 \mathinner{..} t] \cdot [1001111] \mod 2$$
$$y_t^2 := x[t-6 \mathinner{..} t] \cdot [1101101] \mod 2$$
$$\boldsymbol{b}' \leftarrow [y_1^1, y_1^2, y_2^1, y_2^2, \cdots, y_{B+6}^1, y_{B+6}^2]$$

Algorithm 3 実験で使用した畳み込み符号の復号アルゴリズム

Require: $\boldsymbol{q} \in (0,1)^{2(B+6)}$
Ensure: $\tilde{\boldsymbol{q}} \in \{0,1\}^B$ = 復元された元のビット列

$$\phi_0[\boldsymbol{s} \mid \boldsymbol{s} \in \{0,1\}^6] \leftarrow \begin{cases} 0, & \text{if } \boldsymbol{s} = [000000] \\ -\infty, & \text{otherwise} \end{cases}$$
$$\boldsymbol{s}^{\text{prev}}(x, \boldsymbol{s}^{\text{cur}}) := [x] \circ \boldsymbol{s}^{\text{cur}}[1 \mathinner{..} 5]$$
$$o_1(x, \boldsymbol{s}^{\text{cur}}) := ([x] \circ \boldsymbol{s}^{\text{cur}}) \cdot [1001111] \mod 2$$
$$o_2(x, \boldsymbol{s}^{\text{cur}}) := ([x] \circ \boldsymbol{s}^{\text{cur}}) \cdot [1101101] \mod 2$$
$$g(q, b) := b \log q + (1-b) \log(1-q)$$
$$g'(x, \boldsymbol{s}^{\text{cur}}, t) := g(q_{2t-1}, o_1(x, \boldsymbol{s}^{\text{cur}})) + g(q_{2t}, o_2(x, \boldsymbol{s}^{\text{cur}}))$$
$$\phi'(x, \boldsymbol{s}^{\text{cur}}, t) := \phi_{t-1}[\boldsymbol{s}^{\text{prev}}(x, \boldsymbol{s}^{\text{cur}})] + g'(x, \boldsymbol{s}^{\text{cur}}, t)$$

{ 前向き探索 }
for $t = 1 \to B+6$ **do**
  for $\boldsymbol{s}^{\text{cur}} \in \{0,1\}^6$ **do**
    $\hat{x} \leftarrow \arg\max_{x \in \{0,1\}} \phi'(x, \boldsymbol{s}^{\text{cur}}, t)$
    $r_t[\boldsymbol{s}^{\text{cur}}] \leftarrow \boldsymbol{s}^{\text{prev}}(\hat{x}, \boldsymbol{s}^{\text{cur}})$
    $\phi_t[\boldsymbol{s}^{\text{cur}}] \leftarrow \phi'(\hat{x}, \boldsymbol{s}^{\text{cur}}, t)$
  end for
end for
$\boldsymbol{s}' \leftarrow [000000]$
{ 後ろ向き探索 }
for $t = B \to 1$ **do**
  $\boldsymbol{s}' \leftarrow r_{t+6}[\boldsymbol{s}']$
  $\tilde{q}_t \leftarrow s'_1$
end for
$\tilde{\boldsymbol{q}} \leftarrow [\tilde{q}_1, \tilde{q}_2, \cdots, \tilde{q}_B]$

ルゴリズム (Viterbi 1967) に基づく手法であり，畳込み符号を隠れマルコフモデルの一種として解釈し，入力された確率列 $\boldsymbol{q}(\boldsymbol{h})$ から確率的に最も妥当な元のビット列を推定する．Algorithm 3 は一見複雑だが，実際には CPU 上で効率的に処理可能であり，ニューラルネットワークの計算とは分離されているため，翻訳モデル自体の計算効率への影響は避けることが可能である．

表 1 実験に使用したコーパスの詳細.

名称		ASPEC	BTEC
言語対		En ↔ Ja	
文数	Train	2.00 M	465. k
	Dev	1,790	510
	Test	1,812	508
語彙サイズ V		65536	25000

5 実験

5.1 実験設定

提案手法の翻訳精度を比較するために，英日・日英双方向の翻訳タスクでの実験を行った．使用したコーパスは ASPEC (Nakazawa, Yaguchi, Uchimoto, Utiyama, Sumita, Kurohashi, and Isahara 2016) と BTEC (Takezawa 1999) であり，ASPEC は上位 200 万文，BTEC は検証・テスト用以外の全文を学習データとした．表 1 にコーパスの詳細を示す．これらのコーパスは平均文長や語彙サイズの面で難易度が大きく異なるため，難易度に起因するモデルの差異を比較するのに役立つと考えられる．英語のトークン化には Moses (Koehn, Hoang, Birch, Callison-Burch, Federico, Bertoldi, Cowan, Shen, Moran, Zens, Dyer, Bojar, Constantin, and Herbst 2007) に含まれる `tokenizer.perl` を使用し，日本語のトークン化には KyTea (Neubig, Nakata, and Mori 2011) を使用した．また同じく Moses に含まれる `lowercase.perl` により小文字化を行った．語彙は単語の出現頻度に基づいて選択し，出現順位が $V-3$ 位より大きい単語は全て `UNK` 記号に置換した．

ニューラル翻訳モデルを含む全てのアルゴリズムは C++ 言語で記述し，特にニューラルネットワークの構築には DyNet (Neubig, Dyer, Goldberg, Matthews, Ammar, Anastasopoulos, Ballesteros, Chiang, Clothiaux, Cohn, Duh, Faruqui, Gan, Garrette, Ji, Kong, Kuncoro, Kumar, Malaviya, Michel, Oda, Richardson, Saphra, Swayamdipta, and Yin 2017a) を使用した．全てのモデルは 1 個の GPU (NVIDIA GeForce GTX TITAN X) を用いて学習した．テストに関しては，実行時間を検証するために GPU 上と CPU 上の両方で行った．

提案手法が従来のソフトマックスモデルと異なるのは出力層のみであり，他の部分は完全に同一である．ニューラル翻訳モデル全体の構成には，エンコーダに 双方向 RNN (Bahdanau et al. 2014)，注意機構およびデコーダは Luong らによる Concat Global Attention モデル (Luong et al. 2015) を使用した．エンコーダおよびデコーダを構成する RNN には，入力・忘却・出力ゲートを含む 1 層の LSTM (Gers, Schmidhuber, and Cummins 2000) を使用した．また過学習を避けるために，各 RNN の入出力部のみ 30% のドロップアウト（Dropout） (Srivastava,

表 2　比較を行ったモデルの名称と詳細.

名称	概要	図の参照先
Softmax	ソフトマックス（従来手法）	図 2(a)
Adaptive2-N	Head サイズ N の 2 分割適応ソフトマックス（従来手法）	図 3 のソフトマックス化
Binary	単純な二値符号予測	図 2(b)
Hybrid-N	ソフトマックスサイズ N の混合モデル	図 3
Binary-EC	*Binary* の二値符号に畳込み符号を適用	図 2(b), 図 5
Hybrid-N-EC	*Hybrid-N* の二値符号に畳込み符号を適用	図 3, 図 5

Hinton, Krizhevsky, Sutskever, and Salakhutdinov 2014) を導入した．ニューラルネットワークの学習には Adam 最適化器 (Kingma and Ba 2014) を使用した．Adam のハイパーパラメータは $\alpha = 0.001, \beta_1 = 0.9\, \beta_2 = 0.999, \varepsilon = 10^{-8}$ で固定し，文長に基づいてグループ化された 64 文によるミニバッチ学習を行った．各モデルの評価には大文字・小文字を考慮しない BLEU (Papineni, Roukos, Ward, and Zhu 2002) を使用し，ミニバッチ 1000 個の学習が終了するごとにスコアの計算を行った．なお，文中に未知語が存在することを正しく推定したかどうかを評価するために，本実験での BLEU の評価には UNK も計算対象に含めているが，UNK を取り除いて計算した場合でも最終的なスコアの傾向は同様であることを注記しておく．

表 2 に本実験で比較した手法の凡例を示す．特に提案手法については，混合モデルと誤り訂正の適用の有無，および混合モデルにおけるソフトマックス層のサイズ N による影響を実験により検証する．また，混合モデルとの直接比較が可能な 2 分割の場合の適応ソフトマックス (Grave et al. 2017) についても実験を行い，同様に性能の測定を行った．

5.2　実験結果と考察

表 3 に各手法の Test データにおける BLEU，二値符号のビット数 B と出力層で実際に計算される値の個数 $\#_{\text{out}}$，出力層の推定に必要なパラメータ数 $\#_{W,\boldsymbol{\beta}}$，および出力層とモデル全体におけるパラメータ数の *Softmax* との比率を示す．表 4 には英語 → 日本語下での各手法の学習時・テスト時における平均実行時間を示す．また図 6，7 には学習済みミニバッチ数 180,000 までの英日翻訳におけるソフトマックス及び提案手法の Test データ上での BLEU の変化を示す．図 6，7 から明らかなように，学習中の BLEU の変動は不安定であり，比較には何らかの平滑化が必要である．このため，Dev データ上で BLEU が最大となった世代を中心とした 5 世代について Test データ上で BLEU を求め，表 3 にはその平均値を示した．

5.2.1　空間効率

まず，提案手法のいずれにおいても，*Softmax* と比較して大幅に出力層のパラメータ数を削減していることが確認できる．モデル全体のパラメータ数では，いずれの提案手法も *Softmax*

表 3 各手法の BLEU，出力層のパラメータ数とその *Softmax* に対する比率.

コーパス	モデル	BLEU %		B	$\#_{\text{out}}$	$\#_{W,\beta}$	*Softmax* との比率	
		En → Ja	Ja → En				$\#_{W,\beta}$	All
ASPEC	*Softmax*	31.13	21.14	—	65536	33.6 M	1/1	1
	Adaptive2-512	29.96	19.70	—	65537	33.6 M	1/1.00	1.00
	Adaptive2-2048	31.18	21.44	—	65537	33.6 M	1/1.00	1.00
	Binary	13.78	6.953	16	16	8.21 k	1/4.10 k	0.698
	Hybrid-512	22.81	13.95	16	528	271. k	1/124.	0.700
	Hybrid-2048	27.73	16.92	16	2064	1.06 M	1/31.8	0.707
	Binary-EC	25.95	18.02	44	44	22.6 k	1/1.49 k	0.698
	Hybrid-512-EC	29.07	18.66	44	556	285. k	1/118.	0.700
	Hybrid-2048-EC	30.05	19.66	44	2092	1.07 M	1/31.4	0.707
BTEC	*Softmax*	47.72	45.22	—	25000	12.8 M	1/1	1
	Adaptive2-512	45.49	45.03	—	25001	12.8 M	1/1.00	1.00
	Adaptive2-2048	48.33	45.22	—	25001	12.8 M	1/1.00	1.00
	Binary	31.83	31.90	15	15	7.70 k	1/1.67 k	0.738
	Hybrid-512	44.23	43.50	15	527	270. k	1/47.4	0.743
	Hybrid-2048	46.13	45.76	15	2063	1.06 M	1/12.1	0.759
	Binary-EC	44.48	41.21	42	42	21.5 k	1/595.	0.738
	Hybrid-512-EC	47.20	46.52	42	554	284. k	1/45.1	0.744
	Hybrid-2048-EC	48.17	46.58	42	2090	1.07 M	1/12.0	0.760

と比較して 70% 程度のパラメータ数に抑えられており，実質的には従来のソフトマックスモデルで必要とされた出力層のためのパラメータが無視可能なレベルまで削減されたと考えられる．なお，残りのパラメータの大部分はエンコーダおよびデコーダの入力部における単語ベクトルとして確保されているものであり，これらは依然として $O(EV)$ だけのメモリ空間を専有している．ここで E は原言語・目的言語ごとの単語ベクトルに用いられる次元数だが，典型的には H と同程度 (つまり $O(E/H) = O(1)$) と見なすことが可能である．1 節で述べたように，これらのパラメータ数の削減は本研究の対象ではないが，出力層と同様の符号化を導入することは可能であると考えられる．このような手法が翻訳モデルにどう影響するかは今後の研究課題である．また 2.2 節で述べたように，適応ソフトマックスのパラメータ数は基本的に単純なソフトマックスと同等であり，使用するグループ数に比例した分だけ増加している．具体的には，本実験で使用した適応ソフトマックスは分割数が 2 であるため，出力層のサイズは各コーパスの語彙サイズに 1 を足した大きさとなる．

表 4　各手法の平均計算時間.

コーパス	モデル	平均計算時間 (En→Ja) Train [ms/ミニバッチ] (GPU)	Test [ms/文] (GPU)	(CPU)
ASPEC	*Softmax*	1026.	121.6	2539.
	Adaptive2-512	1113.	84.31	884.3
	Adaptive2-2048	1085.	81.17	595.2
	Binary	711.2	73.08	122.3
	Hybrid-512	843.6	81.28	127.5
	Hybrid-2048	837.1	82.28	159.3
	Binary-EC	712.0	78.75	164.0
	Hybrid-512-EC	850.3	80.30	180.2
	Hybrid-2048-EC	851.6	77.83	201.3
BTEC	*Softmax*	325.0	34.35	323.3
	Adaptive2-512	397.4	29.00	129.7
	Adaptive2-2048	387.8	29.19	128.4
	Binary	250.7	27.98	54.62
	Hybrid-512	300.7	28.83	66.13
	Hybrid-2048	307.7	28.25	67.40
	Binary-EC	255.6	28.02	69.76
	Hybrid-512-EC	307.8	28.44	56.98
	Hybrid-2048-EC	311.0	28.47	69.44

5.2.2　翻訳精度

Binary の BLEU 値に着目すると，他のいずれの手法と比較しても大幅に低い値となっていることが観察される．これは 3 節で述べた事実から予測され得る結果であり，単純な二値符号予測では頑健性に問題があるという特徴を反映していると考えられる．これとは対照的に，*Hybrid-N* と *Binary-EC* では *Binary* と比較して十分に高い BLEU を示しており，実験設定によってはほぼ *Softmax* に近似する値を達成していることが分かる．この傾向は，混合モデルと誤り訂正符号の導入がどちらも二値符号予測に対して有効であることを示している．特に *Binary-EC* と *Hybrid-512* を比較すると，前者のパラメータ数が 1/10 程度なのにも関わらず基本的に高い BLEU を達成しており，この結果からビット列に冗長性を導入することがより推定精度の向上に効果的であると言うことができる．また，2 種類の改良を両方導入した *Hybrid-N-EC* では更なる精度の向上が見られ，設定次第では *Softmax* と同等かそれ以上の BLEU を達成していることが観察できる．この挙動から，混合モデルと誤り訂正の両者が手法として概ね直交しており，組み合わせることでより高い効果を示すことが可能であると言える．特に BTEC において， *Softmax* が低い翻訳精度となったのは，コーパスの難易度に対してソフトマックス層のパ

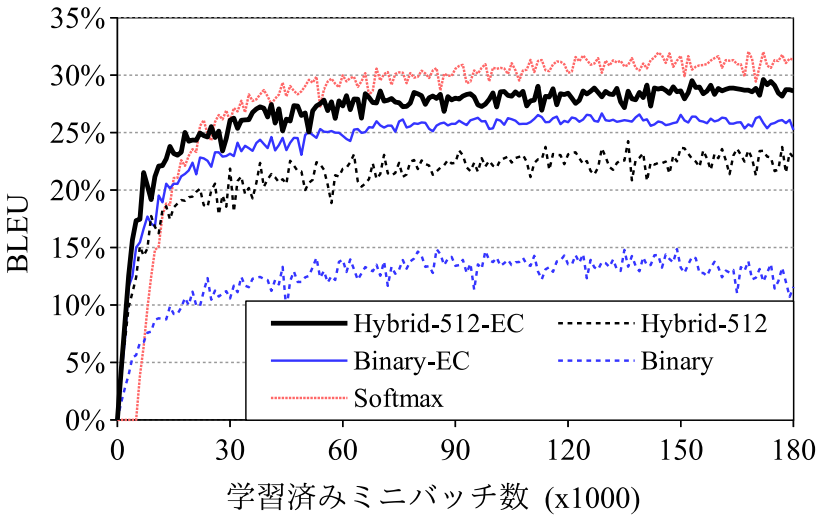

図 6 180,000 世代までの学習の推移 (ASPEC En → Ja).

ラメータ数が適切な大きさでなかった点が考えられる．*Adaptive2-N* と *Hybrid-N-EC* の比較では，ASPEC では *Adaptive2-512* と *Hybrid-2048-EC* が同等程度，BTEC では *Adaptive2-2048* と *Hybrid-2048-EC* が同等程度の性能と考えられる．これらの手法の本質的な違いは希少語グループの単語推定にソフトマックスと誤り訂正符号のどちらを使用したかのみであり，各モデルによる希少語の推定精度を間接的に反映しているものと考えられる．適応ソフトマックスと提案手法の混合モデルでは希少語に関するパラメータ数が数百から数千倍程度異なることを考えると，適応ソフトマックスの翻訳精度が混合モデルより高くなることが自然に考えられ，実際 ASPEC では希少語グループのサイズが同等である *Adaptive2-2048* と *Hybrid-2048-EC* では前者の方が翻訳精度が高いことが観察される．BTEC で両者の性能が接近しているのは，*Softmax* との比較でも言及したように，コーパスの難易度に対する *Adaptive2-N* の過剰なパラメータ数が関係していると考えられる．また，*Adaptive2-2048* はいずれのコーパスでも *Softmax* より高い性能を達成しており，これは混合モデルと同様に，単語の出現頻度に基づいて推定を分割したことによる効果と考えられる．

5.2.3 時間効率

表 4 の計算時間に着目すると，いずれの提案手法も *Softmax* と比較して高速に動作していることが分かる．特に CPU 上でのテスト時の実行速度は ASPEC で 10 倍，BTEC で 5 倍程度に高速であり，強力な計算資源が期待できない環境でも提案手法が効率的に動作可能であることを示している．また，誤り訂正手法の復号アルゴリズムはいずれの実験でも CPU 上で実行し

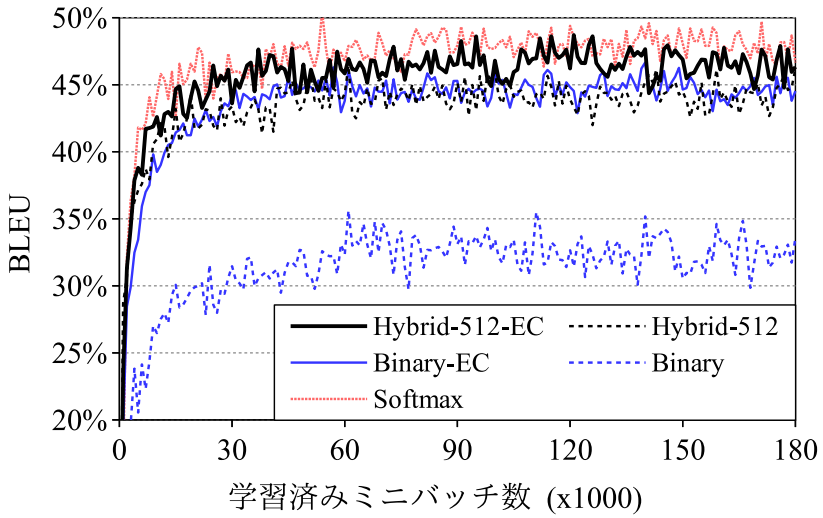

図 7　180,000 世代までの学習の推移 (BTEC En → Ja).

ているが，これに起因する計算速度の低下は翻訳モデル全体から見ると部分的であることが分かり，この点でも誤り訂正手法の適用が効果的であることを示している．*Adaptive2-N* に関しては，GPU 上でのテスト時の実行速度は提案手法と遜色ない性能を示しており，単語の出現頻度に基づいて適切にソフトマックスを分割するだけでも高い高速化効果が得られることを示している．一方，CPU 上でのテスト時の速度は，*Softmax* と比較した際には *Adaptive2-N* でも十分高速化されていると言えるが，提案手法は更にその数割から数倍程度高速という結果となっている．この実行速度の差異は希少語の推定に要する計算量を反映していると考えられ，GPU のように十分な並列化の可能でない状況下では，依然として計算量そのものを削減する利点が大きいことが観察される．なお，*Adaptive2-512* より *Adaptive2-2048* の方が高速に動作している点は，提案論文に示されている 2 分割時の実行速度とのトレードオフに関する議論と一致する (Grave et al. 2017).

ここで，適応ソフトマックス (Grave et al. 2017) の本来の目的は学習時間の削減であるが，本実験では *Adaptive2-N* の学習時間が *Softmax* と比較して若干増加しており，提案論文の結果に従っていないことが確認される．これはミニバッチ学習時の実用上の問題に起因するものである．原理的には，適応ソフトマックスは頻出語の推定時に希少語のスコア計算を除外することが可能であり，このときうまく計算を回避する実装を用いることで計算量の削減が得られるものである．本実験の場合，入力データを個別に処理するテスト時の動作がこれに該当し，実際に期待通りの高速化効果が得られている．一方，学習時には DyNet を用いた実装の制約上，ミニバッチに含まれる全データに対して同様の計算を行う必要があり，ある時刻で推定しなけ

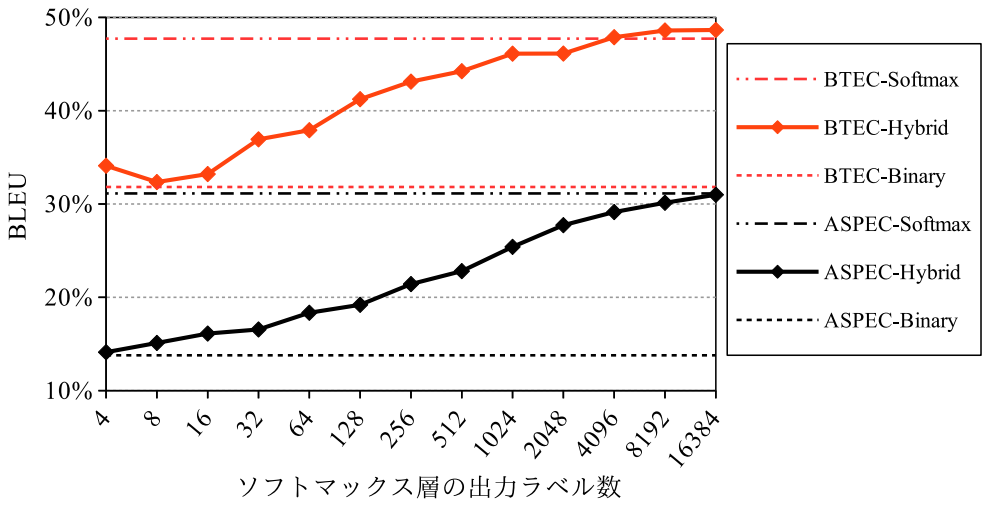

図 8 *Hybrid-N* において N を変化させた場合の BLEU の変化 (En → Ja).

ればならない単語に頻出語と希少語が混在しているような，ネットワークにデータごとの条件分岐が存在する場合への対処が難しい．本実験ではこの問題を回避するため，頻出語に対してもダミーの希少語ラベルを与えて計算を行っているが，このときの計算量は通常のソフトマックスとほぼ等しくなり，また計算過程が複雑になる分，実際の計算効率はソフトマックスよりも低下する．表 4 では実際にこの効果を確認した形となっている．汎用的なニューラルネットワークのツールがこの問題に対処できるかどうかは，データごとに異なるネットワーク構造が与えられたとき，ツール側で効率的に処理する能力を持っているかどうかに依存している．このような手法の候補としては，実際のネットワーク計算の直前にミニバッチ計算の最適化を行う手法が考えられる (Neubig, Goldberg, and Dyer 2017b)．なお，提案手法である混合モデルに関しても，計算グラフ上では適応ソフトマックスとほぼ同様のネットワーク構造を持っており，*Hybrid-N*，*Hybrid-N-EC* の学習時間も上記と同じ理由によるオーバーヘッドを含んでいる．

5.2.4 混合モデルの翻訳精度への影響

図 8 は *Hybrid-N* においてソフトマックス層の大きさ N を指数的に変化させた際の翻訳精度への影響である．ここで混合モデルの定義より，*Softmax* と *Binary* はそれぞれ N を V と 1 に設定した場合の極限であり，*Hybrid-N* による翻訳精度の近似的な上限と下限を表わすと考えられる．実際，図 8 の曲線はおおよそ *Softmax* と *Binary* の間を推移しており，ここから混合モデルのソフトマックス層の大きさと翻訳精度にはトレードオフの関係があることが分かる．また BTEC と ASPEC の曲線をそれぞれ観察すると， BTEC ではより小さな N (例えば

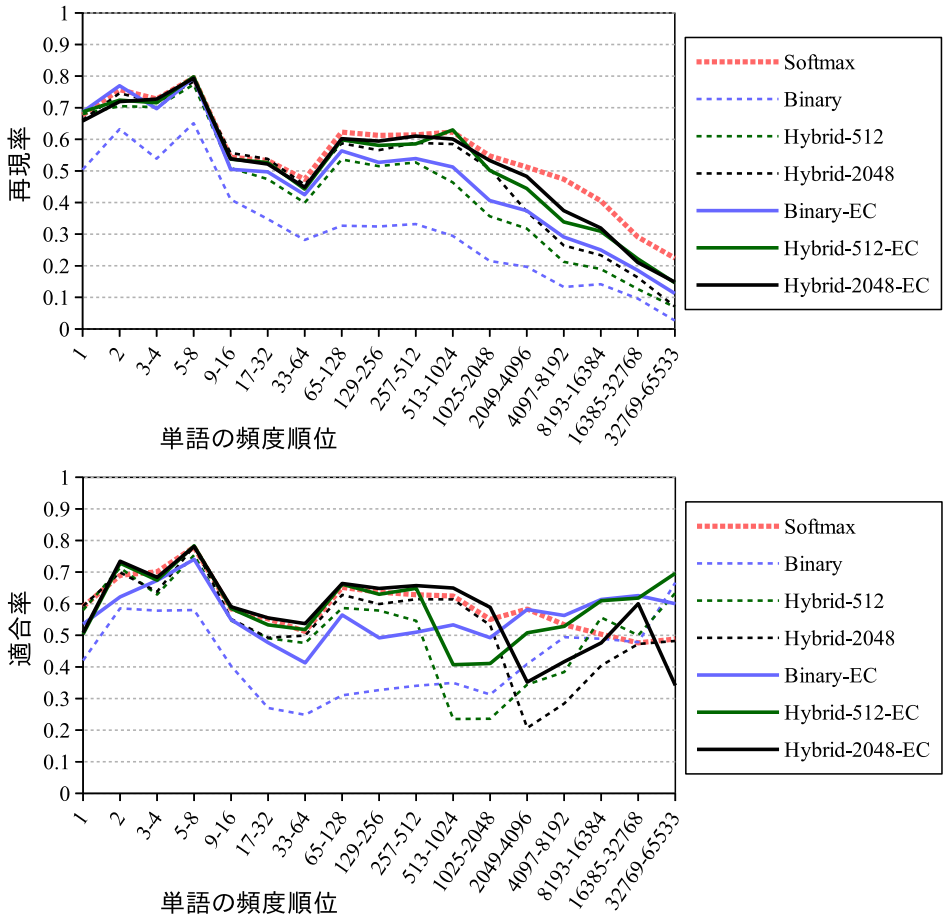

図 9　各手法の 出現頻度ごとのユニグラム推定精度 (ASPEC En → Ja).

$N = 1024$ 程度) で翻訳精度が *Softmax* 付近に飽和しているのに対し，ASPEC ではより大きな N においても精度の向上が見られる．ここから，N による翻訳精度の変動はコーパス自体の難易度をいくらか反映しているものと考えられる．つまり，より簡単なコーパスを学習する場合は N として小さな値を取ることができ，より難しいコーパスになるほど大きな N を選択する必要があると考えられる．また表 3 で示したように，誤り訂正を併用することで混合モデル単体よりも翻訳精度の向上が期待できるため，実際の N の値には図 8 から読み取れるものより小さな値を設定することが可能であると言える．

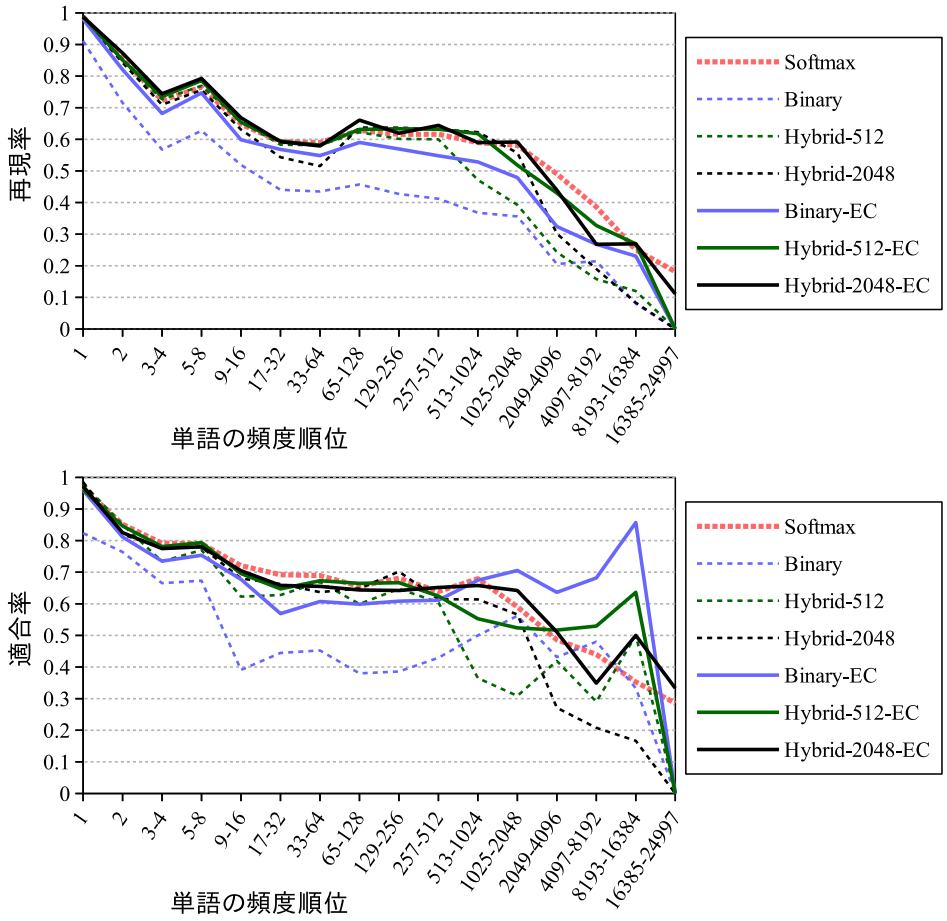

図 10 各手法の 出現頻度ごとのユニグラム推定精度 (BTEC En → Ja).

5.2.5 単語の出現頻度と推定精度の関係

前節までの議論により，提案手法によるコーパス全体としての翻訳精度の傾向については判明した．しかし，ニューラル翻訳は出力文を単語単位で逐次的に生成するモデルであり，具体的な単語単位の推定精度に関しても議論が必要である．図 9，10 に示すのは，表 3，4 に示した 7 手法について，日本語 Test データ中の各単語を学習データ中での出現頻度の順位でグループ化し，各グループについてユニグラム再現率および適合率を示したものである．ここで，適合率の計算は BLEU の方法に準拠し，再現率については BLEU の方法の分母を正解データに置き換えることで計算した．なお，分母が 0 となる場合は便宜的に値を 0 と表示した．

まず，再現率についてはいずれの手法・コーパスにおいてもほぼ同様の傾向を示しており，手法の違いによる全体的な値の増減が確認できる．特に *Binary* のみは，他の手法と比べて全て

表 5　128 位までの頻出語 (ASPEC Ja)

出現順位	単語
1 – 16	の , に を た . し は で る と て が い する な
17 – 31	っ あ ） よ （ 性 さ れ つ 的 こと り から 2 1 も
33 – 64	- 化 く . られ その 用い ため 3 法 行 示 および 結果 例 この 及び 高 ・ 対 中 おけ 細胞 検討 や ／ でき 型 また かっ お ら
65 – 128	など 4 システム 認め う 測定 これ 5 構造 開発 す より 後 体 量 へ 研究 得 解析 率 述べ 年 0 効果 % 可能 変化 方法 値 実験 影響 き 物 特性 群 線 数 系 〜 時間 よう 評価 モデル せ 調べ 間 技術 患者 増加 処理 i 場合 症 症例 治療 内 6 反応 比較 考え 制御 低 機能 時

表 6　128 位までの頻出語 (BTEC Ja)

出現順位	単語
1 – 16	. す い で ま は か の て し を に が た , り
17 – 32	ん な お っ \ る あ せ ください と う よ も 何 だ この
33 – 64	く ね 私 き どこ しょ 行 から どう これ 時 願 一 それ こと み でき 日本 そう まで へ 彼 れ あなた ら する もう さん ここ わ ご 日
65 – 128	つ はい ば さ 人 ござ その いただけ ありがとう 二 や いくら 思 よう 電話 下さ どの 部屋 くれ もの かっ 言 もらえ 時間 予約 いつ 五 来 たら 十 出 くらい わか 見 待 どうぞ 三 ちょっと 車 今 席 彼女 バス 教え ドル 書 食べ 好き なさ かか 持 分 少し ホテル 方 話 じゃ 入 しま られ こちら いいえ 中 ち

のグループで全体的に低い値となっており，単純な二値符号予測ではどのような単語もうまく推定することができないことが分かる．また ASPEC と BTEC の両コーパスで，出現頻度が数十位付近の単語で再現率が低下し，その後また上昇するという共通の傾向を示しており，両コーパスのドメインや難易度が大きく離れていることから，これは言語に共通の何らかの特徴を捉えているものと考えられる．

一般的に単語を出現頻度でグループ化した場合，頻出語のグループには機能語が多く集まり，希少語には内容語が多く集まる傾向にある．内容語と比較して機能語は用例が複雑であり，内容語と同一のモデルで推定するには難易度が高くなると考えられる．表 5 および 6 は ASPEC および BTEC の日本語学習データにおける頻出語だが，両者とも出現頻度が 64 位の付近で機能語と内容語がおおよそ交代していることが観察できる．つまり，前述の Recall の低下と再上昇は，機能語と内容語の推定難易度の違いを反映しているものと考えられる．

一方，適合率の傾向を観察すると，*Binary* が他の手法より全体的に低い点，および誤り訂正の適用により全体的なスコアが引き上げられている点は再現率と同様であるが，*Hybrid-N* において出現頻度の順位が N 付近を超えた単語は，そうでない単語と比較して大幅に適合率が低下

していることが確認できる．これは明らかに，学習の比較的容易なソフトマックス層のみによる推定から二値符号予測を用いる推定に切り替わるために起こる現象であり，二値符号予測が本質的にソフトマックスよりも難しい推定問題であることを表していると考えられる．混合モデルと誤り訂正符号の併用により，このような適合率の大幅な低下が緩和されており，両者をバランス良く組み合わせることで全体的な推定精度を補償することが可能であることが分かる．

6 おわりに

本研究では，ニューラル翻訳モデルの出力層の計算量を圧縮することを目的とし，単語に割り当てられた二値符号を予測することで間接的に単語の推定を行う手法を提案した．また，単純な二値符号予測の精度を向上させるために，従来のソフトマックスモデルを部分的に採用した混合モデル，および二値符号に対し誤り訂正符号を適用することによる頑健性の向上を提案した．実験により，これらの手法を組み合わせて用いることで，従来のソフトマックスモデルと比較して数十分の 1 程度のパラメータ数と短い実行時間（特に CPU 上でのテスト時に 5 分の 1 から 10 分の 1 程度）で，同等程度の翻訳精度を実現可能であることを示した．

本研究では二値符号予測の基本的なフレームワークを提案したが，実験で使用した手法には事前実験に基づく様々なヒューリスティクスが残っており，各部分問題には，より翻訳モデルに適した手法を開発する余地が残されている．具体的には次のような課題が挙げられ，これらについてより深い研究を今後行ってゆく予定である．

- 翻訳モデルにより適した単語のビット列への割り当て手法．より本質的には，どのような特徴量を保持したビット列であれば効率的に予測することが可能か．また，ビット数を制御することで任意に冗長性が設定可能となるような割り当て手法の設計．
- ニューラル翻訳モデルの学習により適した形の誤り訂正手法の開発．また，誤り訂正符号を学習することを前提とした損失関数の設計．
- 入力層側の単語ベクトルも二値符号に制約することで，モデルのパラメータ数をより削減することが可能と考えられるが，そのようなモデルで同様の翻訳精度を達成することは可能か．
- 翻訳モデルの内部状態やパラメータが獲得した表現に関する調査．特に，ソフトマックスモデルと比較した際にどのような共通点・相違点が見出だせるか．

謝 辞

本研究の一部は JSPS 科研費 JP16H05873，及び JP17H00747 の助成を受けて行ったものである．

参考文献

Bahdanau, D., Cho, K., and Bengio, Y. (2014). "Neural machine translation by jointly learning to align and translate." *arXiv preprint arXiv:1409.0473*.

Brown, P. F., Desouza, P. V., Mercer, R. L., Pietra, V. J. D., and Lai, J. C. (1992). "Class-based n-gram models of natural language." *Computational linguistics*, **18** (4), pp. 467–479.

Chen, W., Grangier, D., and Auli, M. (2016a). "Strategies for Training Large Vocabulary Neural Language Models." In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1975–1985, Berlin, Germany. Association for Computational Linguistics.

Chen, Y., Mou, L., Xu, Y., Li, G., and Jin, Z. (2016b). "Compressing Neural Language Models by Sparse Word Representations." In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 226–235, Berlin, Germany. Association for Computational Linguistics.

Chitnis, R. and DeNero, J. (2015). "Variable-Length Word Encodings for Neural Translation Models." In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pp. 2088–2093, Lisbon, Portugal. Association for Computational Linguistics.

Dietterich, T. G. and Bakiri, G. (1995). "Solving multiclass learning problems via error-correcting output codes." *Journal of Artificial Intelligence Research*, **2**, pp. 263–286.

Ferng, C.-S. and Lin, H.-T. (2011). "Multi-label Classification with Error-correcting Codes." *Journal of Machine Learning Research*, **20**, pp. 281–295.

Ferng, C.-S. and Lin, H.-T. (2013). "Multilabel classification using error-correcting codes of hard or soft bits." *IEEE transactions on neural networks and learning systems*, **24** (11), pp. 1888–1900.

Gers, F. A., Schmidhuber, J., and Cummins, F. (2000). "Learning to forget: Continual prediction with LSTM." *Neural computation*, **12** (10), pp. 2451–2471.

Grave, É., Joulin, A., Cissé, M., Grangier, D., and Jégou, H. (2017). "Efficient softmax approximation for GPUs." In *Proceedings of the 34th International Conference on Machine Learning*, pp. 1302–1310, Sydney, Australia.

Gutmann, M. and Hyvärinen, A. (2010). "Noise-contrastive Estimation: A New Estimation Principle for Unnormalized Statistical Models." In *Proceedings of the 13th International Conference on Artificial Intelligence and Statistics*, pp. 297–304, Sardinia, Italy.

Huffman, D. A. (1952). "A Method for the Construction of Minimum-Redundancy Codes." *Proceedings of the Institute of Radio Engineers*, **40** (9), pp. 1098–1101.

Kim, Y. and Rush, A. M. (2016). "Sequence-Level Knowledge Distillation." In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pp. 1317–1327, Austin, Texas. Association for Computational Linguistics.

Kingma, D. and Ba, J. (2014). "Adam: A method for stochastic optimization." *arXiv preprint arXiv:1412.6980*.

Klautau, A., Jevtić, N., and Orlitsky, A. (2003). "On nearest-neighbor error-correcting output codes with application to all-pairs multiclass support vector machines." *Journal of Machine Learning Research*, **4** (April), pp. 1–15.

Koehn, P., Hoang, H., Birch, A., Callison-Burch, C., Federico, M., Bertoldi, N., Cowan, B., Shen, W., Moran, C., Zens, R., Dyer, C., Bojar, O., Constantin, A., and Herbst, E. (2007). "Moses: Open Source Toolkit for Statistical Machine Translation." In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics Companion Volume Proceedings of the Demo and Poster Sessions*, pp. 177–180, Prague, Czech Republic. Association for Computational Linguistics.

Kouzani, A. Z. (2010). "Multilabel classification using error correction codes." In *International Symposium on Intelligence Computation and Applications*, pp. 444–454. Springer.

Kouzani, A. Z. and Nasireding, G. (2009). "Multilabel classification by bch code and random forests." *International journal of recent trends in engineering*, **2** (1), pp. 113–116.

Li, X., Qin, T., Yang, J., and Liu, T.-Y. (2016). "LightRNN: Memory and Computation-Efficient Recurrent Neural Networks." In *Advances in Neural Information Processing Systems*, Vol. 29.

Ling, W., Trancoso, I., Dyer, C., and Black, A. W. (2015). "Character-based neural machine translation." *arXiv preprint arXiv:1511.04586*.

Liu, Y. (2006). "Using SVM and error-correcting codes for multiclass dialog act classification in meeting corpus." In *INTERSPEECH*.

Luong, T., Pham, H., and Manning, C. D. (2015). "Effective Approaches to Attention-based Neural Machine Translation." In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pp. 1412–1421, Lisbon, Portugal. Association for Computational Linguistics.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., and Dean, J. (2013). "Distributed representations of words and phrases and their compositionality." In *Advances in neural information processing systems*, pp. 3111–3119.

Mikolov, T., Sutskever, I., Deoras, A., Le, H.-S., Kombrink, S., and Černocký, J. (2012). "Subword Language Modeling with Neural Networks.".

Mnih, A. and Teh, Y. W. (2012). "A fast and simple algorithm for training neural probabilistic

language models." In *Proceedings of the 29th International Conference on Machine Learning*.

Morin, F. and Bengio, Y. (2005). "Hierarchical Probabilistic Neural Network Language Model." In *Proceedings of Tenth International Workshop on Artificial Intelligence and Statistics*, Vol. 5, pp. 246–252.

Nakazawa, T., Yaguchi, M., Uchimoto, K., Utiyama, M., Sumita, E., Kurohashi, S., and Isahara, H. (2016). "ASPEC: Asian Scientific Paper Excerpt Corpus." In Chair), N. C. C., Choukri, K., Declerck, T., Grobelnik, M., Maegaard, B., Mariani, J., Moreno, A., Odijk, J., and Piperidis, S. (Eds.), *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC 2016)*, pp. 2204–2208, Portoro, Slovenia. European Language Resources Association (ELRA).

Neubig, G., Dyer, C., Goldberg, Y., Matthews, A., Ammar, W., Anastasopoulos, A., Ballesteros, M., Chiang, D., Clothiaux, D., Cohn, T., Duh, K., Faruqui, M., Gan, C., Garrette, D., Ji, Y., Kong, L., Kuncoro, A., Kumar, G., Malaviya, C., Michel, P., Oda, Y., Richardson, M., Saphra, N., Swayamdipta, S., and Yin, P. (2017a). "DyNet: The Dynamic Neural Network Toolkit." *arXiv preprint arXiv:1701.03980*.

Neubig, G., Goldberg, Y., and Dyer, C. (2017b). "On-the-fly Operation Batching in Dynamic Computation Graphs." In *Conference on Neural Information Processing Systems (NIPS)*, Long Beach, California, USA.

Neubig, G., Nakata, Y., and Mori, S. (2011). "Pointwise Prediction for Robust, Adaptable Japanese Morphological Analysis." In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pp. 529–533, Portland, Oregon, USA. Association for Computational Linguistics.

Papineni, K., Roukos, S., Ward, T., and Zhu, W.-J. (2002). "Bleu: a Method for Automatic Evaluation of Machine Translation." In *Proceedings of 40th Annual Meeting of the Association for Computational Linguistics*, pp. 311–318, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

See, A., Luong, M.-T., and Manning, C. D. (2016). "Compression of Neural Machine Translation Models via Pruning." In *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*, pp. 291–301, Berlin, Germany. Association for Computational Linguistics.

Sennrich, R., Haddow, B., and Birch, A. (2016). "Neural Machine Translation of Rare Words with Subword Units." In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1715–1725, Berlin, Germany. Association for Computational Linguistics.

Serrá, J. and Karatzoglou, A. (2017). "Compact Embedding of Binary-coded Inputs and Outputs using Bloom Filters." In *Proceedings of the 5th International Conference on Learning Representations*, Vancouver, BC, Canada.

Shannon, C. E. (1948). "A Mathematical Theory of Communication." *Bell System Technical Journal*, **27** (3), pp. 379–423.

Srivastava, N., Hinton, G. E., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). "Dropout: a simple way to prevent neural networks from overfitting." *Journal of Machine Learning Research*, **15** (1), pp. 1929–1958.

Sutskever, I., Vinyals, O., and Le, Q. V. (2014). "Sequence to sequence learning with neural networks." In *Advances in neural information processing systems*, pp. 3104–3112.

Takezawa, T. (1999). "Building a bilingual travel conversation database for speech translation research." In *Proc. of the 2nd international workshop on East-Asian resources and evaluation conference on language resources and evaluation*, pp. 17–20.

Viterbi, A. (1967). "Error bounds for convolutional codes and an asymptotically optimum decoding algorithm." *IEEE transactions on Information Theory*, **13** (2), pp. 260–269.

Wu, Y., Schuster, M., Chen, Z., Le, Q. V., Norouzi, M., Macherey, W., Krikun, M., Cao, Y., Gao, Q., Macherey, K., Klingner, J., Shah, A., Johnson, M., Liu, X., Kaiser, Ł., Gouws, S., Kato, Y., Kudo, T., Kazawa, H., Stevens, K., Kurian, G., Patil, N., Wang, W., Young, C., Smith, J., Riesa, J., Rudnick, A., Vinyals, O., Corrado, G., Hughes, M., and Dean, J. (2016). "Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation." *arXiv preprint arXiv:1609.08144*.

Zipf, G. K. (1949). *Human behavior and the principle of least effort.* Addison-Wesley Press.

略歴

小田 悠介：2011 年神戸市立工業高等専門学校電子工学科卒業．2013 年同専攻科電気電子工学専攻卒業．2015 年奈良先端科学技術大学院大学情報科学研究科博士前期課程修了．修士（工学）．現在，同博士後期課程に在籍．2015 年日本学術振興会特別研究員（DC1）．2017 年より報通信研究機構先進的音声翻訳研究開発推進センター研究技術員．機械翻訳，自然言語処理，ソフトウェア工学に関する研究に従事．言語処理学会年次大会優秀賞（2015 年・2016 年）．ACL，電子情報通信学会，情報処理学会，教育システム情報学会，言語処理学会各会員．

Philip Arthur：2013 年インドネシア大学計算機科学科卒業．2015 年奈良先端科学技術大学院大学情報科学研究科博士前期課程修了．現在，同学博士後期

課程に在籍．修士（工学）．機械翻訳，自然言語処理に関する研究に従事．

Graham Neubig：2005 年米国イリノイ大学アーバナ・シャンペーン校工学部コンピュータ・サイエンス専攻卒業．2010 年京都大学大学院情報学研究科修士課程修了．2012 年同大学院博士後期課程修了．同年奈良先端科学技術大学院大学助教．2016 年より米国カーネギーメロン大学助教．機械翻訳，自然言語処理に関する研究に従事．

吉野 幸一郎：2009 年慶應義塾大学環境情報学部卒業．2011 年京都大学大学院情報学研究科修士課程修了．2014 年同研究科博士後期課程修了．同年，日本学術振興会特別研究員（PD）．2015 年奈良先端科学技術大学院大学情報科学研究科特任助教．2016 年より同助教．同年より科学技術振興機構さきがけ研究員（兼任）．京都大学博士（情報学）．音声言語処理および自然言語処理，特に音声対話システムに関する研究に従事．2013 年度人工知能学会研究会優秀賞受賞．IEEE，ACL，SIGdial，情報処理学会，言語処理学会各会員．

中村 哲：1981 年京都工芸繊維大学工芸学部電子工学科卒業．京都大学博士（工学）．シャープ株式会社．奈良先端科学技術大学院大学助教授，2000 年 ATR 音声言語コミュニケーション研究所室長，所長，2006 年 (独) 情報通信研究機構研究センター長，けいはんな研究所長などを経て，現在，奈良先端科学技術大学院大学教授．ATR フェロー．カールスルーエ大学客員教授．音声翻訳，音声対話，自然言語処理の研究に従事．情報処理学会喜安記念業績賞，総務大臣表彰，文部科学大臣表彰，Antonio Zampoli 賞受賞．IEEE SLTC 委員，ISCA 理事, IEEE フェロー．