ニューラルかな漢字変換システム Zenzai

三輪敬太 [1] 高橋直希 [2]
[1] 東京大学・Turing 株式会社 [2] 早稲田大学・株式会社 CoeFont
miwakeita@g.ecc.u-tokyo.ac.jp, t-naoki314@akane.waseda.jp

概要

かな漢字変換は日本語話者において広く普及している自然言語処理応用の一つであるが，その精度は未だに十分とは言えない．本研究では，入力のかな文字列に対応する漢字かな交じり文を生成する条件付きニューラル言語モデルを用いたニューラルかな漢字変換システムを提案し，統計的かな漢字変換システムをドラフトモデルとする投機的デコーディングによって提案手法を高速化する．提案手法の精度および速度の検証を行った結果，従来手法に比べて大幅に高い変換精度と，投機的デコーディングの高速化への有効性が示された．

1 はじめに

かな漢字変換システムは音声に対応するかな文字列を入力として受け取り対応する漢字かな交じり文を出力するシステムであり，数千以上の漢字を常用する日本語話者において，最も普及した自然言語処理応用の一つであるといえる．

近年，ニューラルモデルを用いた自然言語処理が急速に発展し，多くのタスクで著しく性能が向上している．かな漢字変換でもニューラル手法による性能向上が示されている [1, 2] ものの，速度・精度のバランスが必要となる実用化に関しては，十分な知見が共有されているとは言いがたい．

本研究では，ニューラルかな漢字変換システム「Zenzai」を提案する．提案システムでは GPT-2 [3] をベースとする条件付き言語モデルによってかな漢字変換をモデル化し，これを統計的かな漢字変換システムをドラフトモデルに用いた投機的デコーディング [4] によって高速化し，精度と速度の両立を図る．提案システムは変換精度で従来手法を凌駕し，一般的なデスクトップ環境で実用的な速度で利用できる．なお，学習データ，評価データ，学習済みモデルおよび提案手法の実装は公開を予定する．

2 背景

かな漢字変換は長く研究されてきた自然言語処理応用の一つであり，1990 年代後半から，統計的かな漢字変換と呼ばれるパラダイムが主流となった [5, 6]．典型的な定式化では，読み y に対する漢字かな交じり文 x の確率 $P(x \mid y)$ を直接求めず，ベイズの定理に基づき $P(x)P(y \mid x)/P(y)$ と分解した上で，分母の定数を無視し，分子の積を最大化する．言語モデル $P(x)$ および「かな漢字モデル」と呼ばれる $P(y \mid x)$ をコーパスから推定し，この確率モデルに基づいて最適な漢字かな交じり文を選択する．Mozc [7] では大規模な Web データを用いて学習したクラス 2 グラムモデルと単語・読みユニグラムモデルを利用してこれを実装している．

近年になり，この $P(x \mid y)$ を End-to-End の条件付きニューラル言語モデルとして直接モデル化するニューラルかな漢字変換が研究されている．これは統計的機械翻訳に対するニューラル機械翻訳 [8] の位置付けと同様であり，RNN に基づく手法 [1] や Transformer ベースの Encoder-Decoder モデルに基づく手法 [2] が提案されているほか，手法の詳細は公開されていないものの，いくつかのプロプライエタリソフトウェアでもニューラルかな漢字変換手法が採用されていると目される [9, 10]．

Transformer [11] ベースのニューラルかな漢字変換は注意機構により文脈全体を考慮した出力が可能であるなどの点で変換精度の向上が期待される一方，推論は従来手法に比して低速である．ローカル環境での即時処理が要求される日本語入力システムでは速度と精度のバランスを取ることが重要となる．

3 提案手法

本研究では，高精度なかな漢字変換のための条件付きニューラル言語モデルを構築し，これを投機的デコーディングに基づく手法で高速化したシステム「Zenzai」を提案する．

3.1 入出力の形式

ニューラルかな漢字変換 [1, 2] の定式化を採用し，かな漢字変換を条件付き言語モデルによってモデル化する．Transformer Decoder ベースの GPT-2 [3] を採用し，条件となる入力から，対応する漢字かな交じり文の出力を自己回帰で生成する．

入力列は，入力トークン $[i_0, i_1, \ldots, i_n]$ と出力トークン $[o_0, o_1, \ldots, o_m]$ を区切りトークン<boi>, <boo>, <eoo>を用いて連結して構成する．この結果，以下のようなトークン列が得られる．モデルは，この列における次トークン予測の交差エントロピー損失を用いて訓練する．

$$S = [\texttt{<boi>}, i_0, \ldots, i_n, \texttt{<boo>}, o_0, \ldots, o_m, \texttt{<eoo>}]$$

入力のみを条件とする変換としては許容されるが，左右の文脈や過去の入力履歴，ユーザの特性などを考慮すると許容されないような候補の提案は，かな漢字変換の典型的な失敗である．このような追加の情報を条件として取る条件付きかな漢字変換の一つとして，入力の左文脈 $\mathbf{c}$ を条件とする**左文脈付きかな漢字変換**をサポートする．左文脈は多くの場面で候補の可能性を狭めることに貢献する有用な情報である．

左文脈付きかな漢字変換では，左文脈と入力を共に条件とする $P(\mathbf{o} \mid \mathbf{i}, \mathbf{c})$ をモデル化する必要がある．この場合のトークン列は次のように，文脈なしのトークン列の先頭に文脈を示すトークン列を追加する形で構成される．これはピンイン・中国語文変換を行う PinyinGPT-Concat [12] と同様の形式である．

$$S = [\texttt{<boc>}, c_0, \ldots, c_l, \texttt{<boi>}, \ldots, \texttt{<boo>}, \ldots, \texttt{<eoo>}]$$

3.2 デコード戦略

デコードには高速かつ決定論的な挙動が得られる貪欲法を採用し，各ステップで最も確率の高い次トークンを選択しトークン列の末尾に追加する．<eoo>トークンが選択された場合，生成を停止する．

かな漢字変換は条件となる入力から出力のほとんどが定まるため，多くのステップで次トークンのエントロピーが低い．そこで高速化のため，投機的デコーディング [4] を適用する．投機的デコーディングは，ターゲットとなる低速なモデルのみを利用して生成するのではなく，高速軽量な「ドラフトモデル」を用いて未来の数ステップ分の候補トークンを一度に提案し，それらをターゲットモデルで並列に受理または修正することで高速な生成を行う手法である．特に貪欲法と共用するケースではブロック単位並列デコーディング [13] と一致し，生成ステップの大部分をドラフトモデルで行いつつ，最終的な出力はターゲットモデルのみの場合と同一になる．

Algorithm 1 投機的デコーディング

1: **Input:** User Input Q
2: **Initialize:** $C \leftarrow$ Null
3: **while** True **do**
4: $\quad A \leftarrow \mathrm{DraftModel}(Q, C)$
5: $\quad C \leftarrow \mathrm{TargetModel}(Q, A)$
6: $\quad$ **if** C = Null **then**
7: $\quad\quad$ **return** A
8: $\quad$ **end if**
9: **end while**

実際の処理手順はアルゴリズム 1 で表現できる．本研究では，ドラフトモデルは入力全体に対する変換候補を提案する．貪欲法のため，受理の判定ではターゲットモデルによって全候補トークンがそれ以前の文脈に対して最も確率の高いトークンであるかを確認する．条件を満たさない候補トークンがあった場合，最大確率のトークンを修正案として，制約 C を更新する．C が Null の場合，ドラフトが受理されたものとして生成を終了する．

通常，投機的デコーディングのドラフトモデルには小規模なニューラル言語モデルを用いるが，本研究では統計的かな漢字変換に基づく軽量な実装を採用する．これは計算速度やメモリ効率の面で大きな利点を持ち，高品質なドラフトを出力できる．

ドラフトモデルに統計的かな漢字変換の実装を採用することには副次的な利点もある．End-to-End の条件付きニューラル言語モデルは直接的な最適化を可能にするが，辞書モジュールを陽に持たないため，ユーザ入力への忠実性を保証しづらい．そこでドラフトモデルに辞書参照を持つシステムを採用し，ターゲットモデルの要求する修正を必ずドラフトモデルで行うことで，最終的な出力がドラフトモデルの利用する辞書項目との明示的な対応を持つようにできる．これによりユーザ入力に忠実でない生成をニューラルモデルが行ったことを検出し，拒否することができる．また，行 3 の繰り返しは任意回実行する代わりに，推論回数制約 L 以下に制限することができる．これは精度の低下を伴うが，計算資源に合わせた負荷の調整を可能にする．

4 実験

学習 モデルサイズとして, GPT-2 medium (310M), small (91M), xsmall (26M) の 3 種類のモデルを構築する．日本語データで事前学習済みのモデル[1]を初期重みとして用いる．ただし，xsmall については事前学習済みモデルが入手できなかったため，ランダムに重みを初期化する．

llm-jp-corpus-v3 [14] および Wikipedia 日本語版アーカイブの一部データに対して MeCab [15] および ipadic-NEologd [16, 17] を用いた読み推定を行い，これを用いて上記の形式のかな漢字変換形式のデータを作成する．学習データは約 190M ペアである．なお，今回構築した学習データは公開を予定する[2]．

文脈，入力および出力のトークナイズには事前学習済みモデルで利用されている Byte-Pair Encoding に基づく Byte-Fallback 付き文字レベルトークナイザを用いる．語彙数は 6000 語である．

GPT-2 の学習には llm.c[3]の実装を用い，1 枚の NVIDIA H100 80GB を利用する．

推論 学習を行ったモデルを llama.cpp[4]を用いて読み込み，端末内蔵の GPU を用いて推論を行う．一般的な実行環境として，Apple M2 Pro 搭載 Mac mini (macOS 14.6.1・メモリ 32 GB) を用いる．どの条件でも KV キャッシュを利用する．

投機的デコーディングのためのドラフトモデルには統計的かな漢字変換システムである AzooKeyKanaKanjiConverter[5]の実装を用いる．

評価 変換の誤りを引き起こしやすい文章における性能評価を目的とし，日本語 Wikipedia 入力誤りデータセット [18] のテストデータのうち，かな漢字変換由来の誤りに分類されている項目を抽出し，ここから 200 件のデータをサンプルした．学習データと同様の方法で読み推定を行ったのち，100 件を通常のかな漢字変換用評価データ，残りの 100 件を左文脈付きかな漢字変換用評価データとするよう，分割した．さらに人手によって読み推定誤りの修正と許容される変換候補のアノテーションを追加し，評価データとした．評価データは公開を予定する．

このデータを用い，次の二つの指標を計算する．

- **Acc@1** 最上位の変換候補が正解であったような評価データの割合として定義する．評価データには複数の正解が存在することがあるため，そのうちの一つに一致すれば正解とみなす．
- **平均文字誤り率（CER）** 文字レベルでの編集距離に基づき計算される誤り率であり，次式で定義する．評価データには複数の正解が存在する場合は，最小の値を代表値とする．

$$\mathrm{CER} = \frac{\sum_{i=1}^{N} \mathrm{EditDistance}(\mathrm{x}_i, \mathrm{y}_i)}{\sum_{i=1}^{N} \mathrm{Length}(\mathrm{x}_i)}$$

ここで，x_i は i 番目の正解文字列，y_i は i 番目の最上位の変換候補を表す．

提案手法に対する統計的かな漢字変換手法として Google 日本語入力と提案手法で用いるドラフトモデルを比較する．また，近年の大規模言語モデルとして GPT-4o と GPT-4o mini [19] も評価する．

精度の評価に加え，推論における投機的デコーディングの有無を制御し，異なるデコード戦略の速度・精度への影響を検討する．このため，評価データの変換の実行にかかる時間を，入力 1 文字あたりの平均で報告する．

これらの評価に関する詳細は Appendix Aに示す．

5 結果

5.1 従来手法との比較

Table 1に各手法の変換精度の評価結果を示す．提案手法は xsmall，small，medium の全てのモデルサイズで統計的かな漢字変換手法である Google 日本語入力とドラフトモデルの結果を大きく上回った．また，変換精度はモデルサイズが大きいほど高く

表 1 変換精度の比較

手法	Acc@1↑	CER↓
Google 日本語入力	54.0	6.7
ドラフトモデル	44.5	7.6
`gpt-4o-2024-08-06`	56.0	7.7
`gpt-4o-mini-2024-07-18`	19.0	21.8
Zenzai (xsmall)	66.5	4.6
Zenzai (small)	80.0	2.6
Zenzai (medium)	**86.5**	**1.7**

1) https://huggingface.co/ku-nlp/gpt2-small-japanese-char より取得.
2) 実験では他システムとの比較を行うが，それぞれのシステムが構築に用いた日本語データは不明であり，これを統制することは不可能であった.
3) https://github.com/karpathy/llm.c より取得．学習には部分的に調整したフォークを用いた.
4) https://github.com/ggerganov/llama.cpp より取得．推論には部分的に調整したフォークを用いた.
5) https://github.com/ensan-hcl/AzooKeyKanaKanjiConverter

なっていた．従来手法が「領主の家に咆哮」と出力した同音異義語の評価項目でも「領主の家に奉公」と正しく変換できるなど良好な性能を示した．

GPT-4o mini は性能が顕著に低かったが，GPT-4o [19] は統計的かな漢字変換手法と同程度の性能を示した．一方で，「欲求の無闇な過熱」を「欲求の無闇な加熱」とするようなエラーなど，多くの日本語話者が解決可能な問題でも多くの失敗が見られた．

5.2 デコード戦略の影響

投機的デコーディング (SD) を異なる設定で行った際の変換精度・速度を Table 2に示す．SD の有無，推論回数制約 $L \in \{1, 2, 3, \infty\}$，モデルサイズを変化させ，15 種類の条件で評価を行った．速度については入力 1 文字あたりの所要時間 (ms) を報告する．

変換速度　結果から，medium サイズと small サイズのモデルでは投機的デコーディングの導入により速度が大きく向上していることがわかる．また，推論回数制約を制御することで，精度劣化を伴うものの，所要時間を大幅に短縮できることがわかる．一方で，xsmall サイズのモデルでは投機的デコーディングの導入でむしろ速度低下が生じることがわかった．ドラフトモデルのみを用いた場合も 1 文字あたり約 1.3ms を必要とすることから，xsmall サイズのモデルが十分に軽量であるために投機的デコーディングの導入による追加の計算コストが高速化の分を上回ったと考えられる．

表 2　異なるデコード戦略を用いた場合の精度と速度

Model	Decoding	Acc@1↑	CER↓	Time↓
xsmall	-SD	**66.5**	5.7	**3.5**
	+SD, L=∞	**66.5**	**4.6**	6.1
	+SD, L=3	65.0	5.4	5.4
	+SD, L=2	65.5	6.0	5.0
	+SD, L=1	61.5	5.7	3.7
small	-SD	**80.5**	3.4	9.8
	+SD, L=∞	80.0	**2.6**	6.0
	+SD, L=3	77.0	3.3	5.8
	+SD, L=2	75.5	3.8	5.1
	+SD, L=1	69.5	3.9	**4.2**
medium	-SD	**86.5**	2.8	26.4
	+SD, L=∞	**86.5**	**1.7**	6.9
	+SD, L=3	83.0	2.1	5.8
	+SD, L=2	79.0	2.9	6.2
	+SD, L=1	70.5	3.6	**4.8**

変換精度　投機的デコーディングでは変換精度は同等であることが期待される．xsmall サイズと medium サイズのモデルでは-SD 条件と+SD,L=∞ 条件で同等の Acc@1 が確認された．しかし，small サイズのモデルでは-SD 条件の方が Acc@1 で 0.5 ポイント高かった．これは-SD 条件で正しく変換できた「夜駆け（ヨガケ）」という単語を「駆け=ガケ」を辞書に持っていないドラフトモデル側が拒否してしまったことに由来する．このような挙動は複数見られ，xsmall モデルの-SD 条件で「センゴハダカイッカンカラ」を「戦後は蛇行一貫から」としたところを，+SD 条件では読みに忠実に「戦後は蛇か一環から」とした例があった．-SD 条件と+SD, L=∞ 条件の CER の差異については，-SD 条件でモデルの生成が繰り返しに陥ってしまうエラーが発生し極度に大きな CER が観測される評価データが存在することが大きく影響したものであり，それ以外の点で CER に大きな違いは見られなかった．

なお，推論回数制約を小さくした際の精度はドラフトモデルの性能に依存する．$L = 1$ の条件で，small サイズと medium サイズの間の性能差が非常に小さいのは，ドラフトモデルの出力のうち 1 回以下の修正で正解に出来るものが最大 70%程度であることを示唆する．このため，ドラフトモデルの性能がより高ければ必要な修正回数が減り，より強い推論回数制約でより高い性能を実現できる．この結果から，本手法ではドラフトモデルに使用するかな漢字変換手法の性能向上が依然重要だといえる．

6 おわりに

本研究ではニューラルかな漢字変換システム「Zenzai」を提案し，大幅な精度向上を実現した．また，投機的デコーディングを用いることで，提案システムが高速化できることを示した．

日本語入力システムは日本語話者において最も普及している自然言語処理応用の一つであり，最先端の技術を用いて改善されるべき余地が広く残されている．本稿では紙面の都合から触れられなかったが，履歴学習，ユーザ辞書，プロンプト条件付き変換などの個人最適化や，予測入力などの入力支援技術についても開発に取り組み，近年の知見の導入を進めている．Zenzai はすでに macOS を対象としたオープンソースの IME[6] に搭載されて実用化されており，本稿は全てこれを用いて執筆された．

6）https://github.com/ensan-hcl/azooKey-Desktop

謝辞

本研究にあたり，独立行政法人情報処理推進機構（IPA）2024 年度未踏 IT 人材発掘・育成事業の支援を受けました．ここに深く謝意を表します．

また，モデルの開発・学習にあたり，さくらインターネット株式会社様に計算資源の支援をいただきました．この場を借りて深く感謝いたします．

本研究の評価データ作成にご協力いただいた堀山峻生さん，村山裕基さん，和内直也さんに感謝します．

参考文献

[1] Yoh Okuno. Neural IME: Neural input method engine. Presented at The 8th Input Method Workshop, 2016.

[2] Armin Sarhangzadeh and Taro Watanabe. Alignment-based decoding policy for low-latency and anticipation-free neural Japanese input method editors. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar, editors, **Findings of the Association for Computational Linguistics: ACL 2024**, pp. 8043–8054, Bangkok, Thailand, August 2024. Association for Computational Linguistics.

[3] Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. 2019.

[4] Yaniv Leviathan, Matan Kalman, and Yossi Matias. Fast inference from transformers via speculative decoding. In **Proceedings of the 40th International Conference on Machine Learning**, ICML'23. JMLR.org, 2023.

[5] 森信介, 土屋雅稔, 山地治, 長尾真. 確率的モデルによる仮名漢字変換. 情報処理学会研究報告. NL, 自然言語処理研究会報告, Vol. 125, pp. 93–99, 05 1998.

[6] 小町守, 森信介, 徳永拓之. あいまいな日本語のかな漢字変換. 情報処理学会夏のプログラミング・シンポジウム, 2008.

[7] 工藤拓, 小松弘幸, 花岡俊行, 向井淳, 田畑悠介. 統計的かな漢字変換システム mozc. 言語処理学会第 17 回年次大会, 2011.

[8] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. In Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K.Q. Weinberger, editors, **Advances in Neural Information Processing Systems**, Vol. 27. Curran Associates, Inc., 2014.

[9] Apple Inc. macos catalina - features, 2019. Accessed: 2025-01-04.

[10] JustSystems. Atok features - engine, 2025. Accessed: 2025-01-04.

[11] A Vaswani. Attention is all you need. **Advances in Neural Information Processing Systems**, 2017.

[12] Minghuan Tan, Yong Dai, Duyu Tang, Zhangyin Feng, Guoping Huang, Jing Jiang, Jiwei Li, and Shuming Shi. Exploring and adapting Chinese GPT to Pinyin input method. In Smaranda Muresan, Preslav Nakov, and Aline Villavicencio, editors, **Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)**, pp. 1899–1909, Dublin, Ireland, May 2022. Association for Computational Linguistics.

[13] Mitchell Stern, Noam Shazeer, and Jakob Uszkoreit. Blockwise parallel decoding for deep autoregressive models. **Advances in Neural Information Processing Systems**, Vol. 31, , 2018.

[14] Research National Institute of Informatics and Development Center for Large Language Models (LLMC). Llm-jp corpus v3, 2024. Accessed: 2025-01-04.

[15] Taku Kudo, Kaoru Yamamoto, and Yuji Matsumoto. Applying conditional random fields to Japanese morphological analysis. In Dekang Lin and Dekai Wu, editors, **Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing**, pp. 230–237, Barcelona, Spain, July 2004. Association for Computational Linguistics.

[16] 佐藤敏紀, 橋本泰一, 奥村学. 単語分かち書き用辞書生成システム neologd の運用 — 文書分類を例にして —. 自然言語処理研究会研究報告, pp. NL–229–15. 情報処理学会, 2016.

[17] 佐藤敏紀, 橋本泰一, 奥村学. 単語分かち書き辞書 mecab-ipadic-neologd の実装と情報検索における効果的な使用方法の検討. 言語処理学会第 23 回年次大会 (NLP2017), pp. NLP2017–B6–1. 言語処理学会, 2017.

[18] Yu Tanaka, Yugo Murawaki, Daisuke Kawahara, and Sadao Kurohashi. Building a Japanese typo dataset from Wikipedia's revision history. In Shruti Rijhwani, Jiangming Liu, Yizhong Wang, and Rotem Dror, editors, **Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: Student Research Workshop**, pp. 230–236, Online, July 2020. Association for Computational Linguistics.

[19] Aaron Hurst, Adam Lerer, Adam P Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, et al. Gpt-4o system card. **arXiv preprint arXiv:2410.21276**, 2024.

A　評価の詳細

本研究では，かな漢字変換エンジンとしての GPT-4o および GPT-4o mini の性能を評価した．解析を容易にするため，Structured Output を用いた．また，文の途中で変換が行われるケースが不得意な傾向があったため，1-shot 例としてそのようなサンプルを示した．以下に実際に使用したプロンプトを示す．

```
{
  "role": "system",
   "content": "あなたはかな漢字変換エンジンとして振る舞います。ユーザの入力に対して JSON で「\"output\": \"...\"」の形でかな漢字変換結果を返してください。"
},
{
  "role": "user",
  "content": "かな漢字変換してください。出力のみを答えてください。\n 文脈：彼は走ろ\n 入力：ウトアシニチカラヲイレタ\n 出力：うと足に力を入れた\n 文脈：<文脈をここに挿入>\n 入力：<入力をカタカナでここに挿入>\n 出力："
}
```

Google 日本語入力については，2025 年 1 月 9 日に Google CGI API for Japanese Input (https://www.google.co.jp/ime/cgiapi.html) にリクエストを送信し，この結果を用いた．ただし，Google CGI API for Japanese Input では長い入力の変換で適切な変換結果を得られないケースがあったため，句読点や括弧を基準に文字列を分割した上で，それぞれを変換し，結合した結果を評価対象とした．

ドラフトモデルと Google CGI API for Japanese Input については，文脈を指定する仕組みが存在しないため，文脈情報は入力せずに評価した．なお，文脈なし項目のみで評価した場合も，提案手法は既存手法を大きく上回った（Table 3）．

表 3　文脈なし項目のみの変換精度の比較

手法	Acc@1↑	CER↓
Google 日本語入力	62.0	4.4
ドラフトモデル	46.0	6.6
gpt-4o-2024-08-06	52.0	9.7
gpt-4o-mini-2024-07-18	18.0	22.7
Zenzai (xsmall)	69.0	4.1
Zenzai (small)	83.0	**1.7**
Zenzai (medium)	**89.0**	1.8

zenz のモデルについては，llama.cpp を用いて Q5_K_M 量子化を施して評価を行っている．量子化後のパラメタサイズは medium サイズで 237.2 MB，small サイズで 72.3 MB，xsmall サイズで 19.9 MB だった．

最後に，評価データのサンプルを Table 4に示す．「ユリ」のような文脈内の固有名詞を用いて変換を行うことが求められる問題や，「領主」と「奉公」のような共起関係にある情報を認識して変換するべき問題が含まれる．

表 4　評価データのサンプル

Index	Item
118	**Context**: 配偶者の氏 **Input**: ニヘンコウ、アルイハフクゴウセイヲセンタクスルタメニハソノヨウニコンインマエニテツヅキヲオコナワ **Possible Outputs**: に変更、あるいは複合姓を選択するためにはそのように婚姻前に手続きを行わ に変更、あるいは複合姓を選択するためにはそのように婚姻前に手続きをおこなわ
2021	**Context**: この設定でもまだユリとは一緒になってい **Input**: ナイラシイ（タイセツナモノニユリヲアゲ **Possible Outputs**: ないらしい（大切なものにユリを挙げ　ないらしい（大切なものにユリをあげ ないらしい（大切なモノにユリを挙げ　ないらしい（大切なモノにユリをあげ ないらしい（大切な物にユリを挙げ　ないらしい（大切な物にユリをあげ
200	**Context**: N/A **Input**: ケンキュウテーマデアルタヘンスウフクソカンスウロン **Possible Outputs**: の研究テーマである多変数複素関数論　の研究テーマである多変数複素函数論
1890	**Context**: N/A **Input**: カケイヲタスケルタメ、リョウシュノイエニホウコウシ **Possible Outputs**: 家計を助けるため、領主の家に奉公し