日本語文平易化のための疑似パラレルコーパス構築

澤柳 翔太 [1]　小川 泰弘 [2]　外山 勝彦 [1]
[1] 名古屋大学大学院情報学研究科　[2] 名古屋市立大学 データサイエンス学部
sawayanagi.shota.r8@s.mail.nagoya-u.ac.jp　ogawa@ds.nagoya-cu.ac.jp
toyama@is.nagoya-u.ac.jp

概要

本研究では, 単言語コーパスをもとに日本語文平易化コーパスを疑似的に構築する手法を提案する. 日本語文平易化タスクにおいては, 大規模なコーパスの不足が課題である. 本研究では, 単言語コーパスから類似度が高く, かつ難易度が異なる 2 文を抽出することにより, 約 6 万ペアを含む疑似日本語文平易化コーパスを構築した. また, 構築したコーパスを用いて平易化モデルを構築することにより, コーパスの品質を評価した. その結果, 提案手法の有効性を明らかにした.

1　はじめに

文平易化は, 文意を保持しつつ難解な文を平易に書き換えるタスクであり, 子供や高齢者, 言語学習者などに対する情報アクセスの向上に寄与する.

英語では, ニュース記事をプロのライターが平易化した Newsela [1] や, Wikipedia とその簡易版である Simple Wikipedia をアライメントしたコーパス [2] など, 難解文と平易文が対になっている平易化コーパスが整備されており, 平易化研究も盛んである. 一方, 日本語では既存の平易化コーパスとして, SNOW T15 [3] や SNOW T23 [4], MATCHA [5] などが挙げられるが, いずれも規模は数万ペア程度に限られている. そのため, 日本語における平易化研究は発展途上であり, 大規模かつ多様な平易化コーパスの構築が求められる.

本研究の目的は, 大規模な平易化コーパスを構築し, 平易化研究における言語資源不足を補うことである. 言語資源不足への対処法として, 単言語コーパスから類似度の高い文ペアを抽出し, 平易化コーパスを疑似的に構築する手法 [6] がある. しかし, その手法には文ペア抽出の効率や品質に課題があると考えられるため, 本研究では, そのアプローチを参考に, 新たな疑似コーパス構築手法を提案する.

本手法では, 文の表層的な類似度と意味的な類似度を併用することにより, 大規模な単言語コーパスから高品質なデータを効率よく抽出することを目指す. 作成した疑似コーパスにより平易化モデルを構築する実験では, 既存の高品質な平易化コーパスを用いた場合に匹敵する平易化スコアを達成し, 提案手法の有効性を示した.

2　関連研究

本節では, 日本語平易化コーパス構築に関する研究を紹介する.

2.1　日本語テキスト平易化コーパス

日本語における代表的な平易化コーパスとして, SNOW T15 [3] や T23 [4], MATCHA [5] が挙げられる.

SNOW T15, T23 は, 独自に定義された「やさしい日本語」表現をもとに, 文を人手で平易化して作成されたコーパスである. 学生により約 5 万ペアを含む T15 が作成され, その後クラウドワーカーにより約 3 万 5 千ペアを含む T23 が作成された.

MATCHA は, 訪日観光客向けメディア「MATCHA」[1] の記事から作成された 1 万 6 千ペアからなるコーパスであり, 文アライメントは専門家により人手で行われた. 定性的評価の結果, コーパス中の文ペアは流暢かつ多様な平易化操作を含んでいることが報告されている.

これらのコーパスは, 日本語平易化研究のリソースとしてよく利用されるが, いずれも規模は十分ではなく, 大規模で多様なデータセットの構築が求められる. 本研究では, 単言語コーパスを活用することでより大規模な平易化データの作成を目指す.

2.2　疑似平易化コーパス構築

既存の平易化コーパスの量的な不足を補うため, 梶原ら [6] は単言語コーパスから平易化文ペアを疑

1) https://matcha-jp.com/

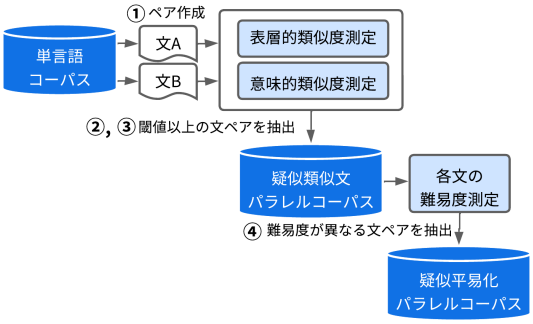

図 1　疑似平易化コーパス構築の流れ

似的に構築する手法を提案している.

この手法では,「意味的に類似し, かつ難易度が異なる 2 文」が平易化文ペアであるとみなし, 単言語コーパスを「標準文」と「平易文」に分類した後, それらの文の間で単語ベースの意味的類似度を計算することにより, 疑似平易化文ペアを抽出する. 実験では, 現代日本語書き言葉均衡コーパス（BCCWJ）[2] から疑似平易化文ペアを抽出し, それらを SMT の学習に用いた結果, 一定の平易化性能を持つモデルが構築可能であると示されている.

しかし, この手法には,「標準文」と「平易文」への分類後, 非類似文ペアを含むすべての文ペアに対して高コストな意味的類似度の計算を行うため, 単言語コーパスが大規模になるほど計算時間が増大するという課題がある.

本研究で提案する手法は, 表層的な特徴に基づく高速な類似度フィルタリングを先行して適用することにより非類似文ペアを効率的に除外し, その後, 意味を考慮した類似度フィルタリングにより高品質な文ペアを抽出するアプローチを採用する. これにより, 既存手法が抱える課題を克服することを目指す.

3　提案手法

本節では, 提案手法の枠組みとその各ステップについて説明する. 本研究では, 梶原ら [6] と同様に, 意味的に類似し, かつ難易度が異なる文ペアを平易化文ペアとみなし, 単言語コーパスから平易化文ペアを効率的に抽出する手法を提案する.

3.1　フレームワークの概要

提案手法の全体の流れを図 1 に示す. 提案手法は以下の 4 つのステップで構成される. なお, その前後に, 文の前処理および後処理も実施している. それらの詳細は付録 A で述べる.

1. 単言語コーパス中の文から文ペアを作成する.
2. 高速な表層的類似度フィルタリングにより非類似文ペアを除外する.
3. 意味的類似度フィルタリングにより意味的に対応する文を抽出する.
4. 文の難易度を測定し, 難易度が異なる類似文ペアのみを抽出する.

3.2　単言語コーパス

単言語コーパスから作成する文ペアのうち, 意味的に対応しており, かつ難易度が異なるものはわずかであると考えられるため, ソースとなる単言語コーパスは, 可能な限り大規模であることが望ましい. 梶原ら [6] が実験に用いた BCCWJ の文数が約 400 万文であったのに対して, 本手法では, 約 40 億文を含む CC100 コーパス [7, 8] の日本語部分を使用する. CC100 は, ウェブ上の膨大なデータをクローリングして作成された多言語コーパスである. ただし, 計算機環境の都合上, 全データを用いることが難しいため, 本実験では 500 万文をランダムサンプリングした. なお, この取得サンプル数を増やせば疑似コーパスの規模を容易に拡張できる.

3.3　表層的類似度フィルタリング

単言語コーパス内のすべての文ペアを意味的に比較することは計算コストが高いため, 高速に計算可能な表層的類似度フィルタリングを先行して適用する. 類似度測定には, Simstring [9] を用いた. Simstring は高速な類似文字列検索のための手法であり, 文字 n-gram の一致度に基づいて, 文字列集合からクエリ文字列との類似度が閾値以上である部分文字列集合を抽出する. 閾値が高いほど表層的に類似する文が得られるが, 文ペアの多様性を確保するため, 本実験では閾値を中程度の 0.5 に設定した.

また, 文ペアの多様性を確保するために, 非類似文ペアの他に, 過剰に類似する文ペアも除外する. Simstring 適用後に残った文ペアに対して, 最大文長で正規化した文字単位の編集距離を計算し, その値が 0.2 未満の文ペアを除外する.

3.4　意味的類似度フィルタリング

文全体の意味的な類似度を測定するために, BERT[3] を使用し, 文埋め込みのコサイン類似度が 0.9 以上の文ペアを抽出する. しかし, BERT のみで

2）https://clrd.ninjal.ac.jp/bccwj/

3）https://github.com/cl-tohoku/bert-japanese

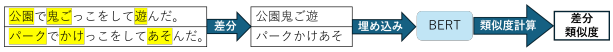

図 2　文字ベースの差分類似度測定

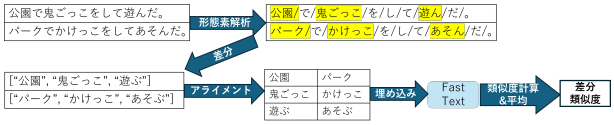

図 3　形態素ベースの差分類似度測定

は, 地名などの固有表現のみが異なる文ペアも抽出してしまうことを確認している. そのような文ペアは平易化文ペアとすべきではないと判断し, BERT に加えて, 文ペアの差分のみの類似度（差分類似度）も測定する. 差分類似度は文字ベースと形態素ベースの 2 種類を併用し, それぞれ以下の方法で実装した.

文字ベース差分類似度測定

差分を文字単位で求め, それを単純に結合した文字列の BERT 埋め込みを取得し, コサイン類似度を計算する. 測定時の流れを図 2 に示す.

形態素ベース差分類似度測定

文を MeCab により形態素解析し, 両文で異なっている形態素のみでアライメントを取り, 対応する形態素間の埋め込みの類似度の平均を計算する. 形態素の埋め込みには Wikipedia で事前学習した FastText [10] を使用し, アライメントには梶原ら [6] の最大アライメントアルゴリズムを用いた. 測定時の流れを図 3 に示す.

2 種類の差分類似度測定で, ともに値が上位 10% であった文ペアを抽出した. 閾値を類似度の値ではなく, 割合で指定する理由は, BERT 埋め込みと FastText 埋め込みで類似度分布が異なるためである.

3.5　文難易度フィルタリング

前節で抽出された各文に難易度ラベルを付与し, 難易度の異なる文ペアを選定することにより, 疑似平易化コーパスを構築する. 難易度測定には澤柳ら [11] の難易度測定モデルを用いた. これは, 難易度順に並び替えた BCCWJ を学習データとして BERT を fine-tuning したモデルであり, 文の難易度を 0 から 8 の 9 段階で分類する. 難易度を多クラスで測定することにより, 幅広い難易度帯から相対的に難易度が異なる文ペアを抽出し, 平易化の多様なケースに対応することができる.

表 1　日本語 T5 のハイパーパラメータ

パラメータ	値
epoch	5
batch size	8
max input length	128
max target length	128

表 2　疑似平易化コーパス：ペア数

設定	ペア数
Pseudo	5,732,115
Pseudo w/ DiffSim	66,860

4　実験

本節では, 提案手法により疑似平易化コーパスを構築し, 平易化モデルを構築した実験について述べる. また, コーパスの品質評価のために, 既存の平易化コーパスを用いたモデルと比較する.

4.1　実験設定

本実験の設定を以下に示す.

平易化モデル

本実験では, 事前学習済み日本語 T5 モデル[4] を平易化モデルとして使用した. このモデルを平易化タスクに特化させるために, 疑似コーパスまたは既存コーパスを用いて fine-tuning した. なお, fine-tuning 時のハイパーパラメータは, すべての設定において, 表 1 に示す値に統一した.

コーパス

fine-tuning に用いる疑似コーパスとして, 3.4 節で述べた差分類似度を適用しない場合（Pseudo）と適用した場合（Pseudo w/ DiffSim）の 2 パターンを作成した. 表 2 に, 疑似コーパスの文ペア数を示す.

また, 比較対象として, 既存の平易化コーパスである T23 と MATCHA のデータでも fine-tuning を実施した. データの抽出法については付録 B で述べる.

疑似コーパスおよび既存コーパスの fine-tuning 用データの数を, 表 3 第 2 列に示す. 2 パターンの疑似コーパスについては, データ数に差があるため, データ数が等しくなるようアンダーサンプリングした.

4）https://huggingface.co/sonoisa/t5-base-japanese

表 3 データ数および fine-tuning 後の SARI スコア

設定	データ数	SARI	
		T23	MATCHA
Pseudo	66,860	0.311	0.298
Pseudo w/ DiffSim	66,860	0.312	0.332
T23	29,300	0.650	0.413
MATCHA	6,301	0.354	0.590

評価指標

モデルの性能評価のために, 平易化タスクで広く用いられる指標である SARI スコア [12] を計算した. スコアは 0 から 1 の間で計算され, 高いほど平易化の品質が高いと評価される. 評価データとして T23 から 5,000 ペア, MATCHA から 1,075 ペアを, それぞれ fine-tuning 用データとは独立に抽出した.

4.2 結果

各設定で T5 モデルを fine-tuning した結果を表 3 に示す.「SARI」の列は, T23 評価データと MATCHA 評価データそれぞれにおける SARI スコアである. なお, 各モデルにおける出力例は付録 C に示す.

まず, 疑似コーパス間を比較すると, 両方の評価データにおいて差分類似度の適用あり（Pseudo w/ DiffSim）のスコアの方が高い. 特に, MATCHA 評価データで 3.4 ポイントの差があることが確認できた.

次に, 既存コーパス間を比較すると, 学習データと評価データのコーパスが一致している場合に SARI スコアが高く, そうでない場合にスコアが大きく低下していることが確認できた.

最後に, 既存コーパスと疑似コーパスを比較すると, すべての設定において既存コーパスの SARI スコアの方が高い. ただし, MATCHA で fine-tuning し, T23 で評価した際のスコア 0.354 は, 疑似コーパスの場合と近いスコアになっている.

4.3 考察

以下に, 実験結果に対する考察を述べる.

差分類似度の効果

疑似コーパスにおいて差分類似度を導入することにより SARI スコアが改善した理由は, 固有表現など特定の部分のみが異なる文ペアを除外し, より意味的に対応した高品質な文ペアが抽出できたためと考えられる. 特に, MATCHA は観光客向けの文を多く収録しており, 地名などの固有表現が多い. 差分類似度を適用しない場合, それらを別の固有表現に言い換えてしまい, スコアが低くなったと考えられる.

既存コーパスどうしの比較

既存コーパスにおいて, 学習時と評価時のコーパスが異なる場合に SARI スコアは大きく低下した. これは, 両コーパスの性質の違いが関係していると考えられる. T23 では主な平易化操作が単語の置換である. 一方, MATCHA は語句の挿入や並び替えなど様々な平易化操作を含む. この違いから, モデルの平易化傾向が異なると考えられる. また, 両コーパスのサイズの都合上, fine-tuning 用データの数が異なることもスコアの違いに影響していると考えられる.

疑似コーパスの有用性

疑似コーパスで学習した設定が, 既存コーパスで学習した設定と近いスコアを示したことから, 提案手法が日本語平易化タスクにおける新たなデータ作成の手段として有用であることが示された. また, 本手法は, 単言語コーパスのサイズを拡張すれば, より多くの平易化文ペアを構築可能であり, さらにスコアを改善できる可能性もある.

また, 既存コーパスのうち, 特に MATCHA とスコアが近かった理由は, 疑似コーパスと MATCHA がともにアライメントを基盤とするアプローチによりコーパスを構築しているためと考えられる. MATCHA は, 元記事を人手でアライメントすることで文ペアを作成している. 一方, T23 は所与の文をクラウドソーシングによって言い換え, 文ペアを「生成」する形で作成している. そのため, MATCHA の方が疑似コーパスの特性と類似しており, スコアが近くなったと考えられる.

5 おわりに

本研究では, 日本語平易化のための疑似平易化コーパスを構築する手法を提案した. 提案手法は, 単言語コーパスをもとに, 表層的および意味的類似度によるフィルタリングを適用し, 効率的に難解文と平易文のペアを抽出するアプローチである. 構築した疑似コーパスを用いて平易化モデルを学習する実験により, 提案手法の有効性を示すことができた.

今後の課題として, 疑似コーパスの品質向上が挙げられる. 構築したコーパス内には非類似文ペアがいまだ多く含まれており, それらを除去するためのさらなるフィルタリング手法の検討が必要である.

謝辞

本研究はJSPS科研費JP23K25155の助成を受けた.

参考文献

[1] Xu Wei and Callison-Burch. Problems in current text simplification research: New data can help. **Transactions of the Association for Computational Linguistics**, Vol. 3, pp. 283–297, 2015.

[2] Xingxing Zhang and Mirella Lapata. Sentence simplification with deep reinforcement learning. **Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing**, pp. 584–594, 2017.

[3] Takumi Maruyama and Kazuhide Yamamoto. Simplified corpus with core vocabulary. **Proceedings of the 11th International Conference on Language Resources and Evaluation**, pp. 1153–1160, 2018.

[4] Akihiro Katsuta and Kazuhide Yamamoto. Crowdsourced corpus of sentence simplification with core vocabulary. **Proceedings of the 11th International Conference on Language Resources and Evaluation**, pp. 461–466, 2018.

[5] 宮田莉奈, 惟高日向, 山内洋輝, 柳本大輝, 梶原智之, 二宮崇, 西脇靖紘. Matcha：専門家が平易化した記事を用いたやさしい日本語パラレルコーパス. 自然言語処理, Vol. 31, No. 2, pp. 590–609, 2024.

[6] 梶原智之, 小町守. 平易なコーパスを用いないテキスト平易化. 自然言語処理, Vol. 25, No. 2, pp. 223–249, 2018.

[7] Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. Unsupervised cross-lingual representation learning at scale. **Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics**, pp. 8440–8451, 2020.

[8] Guillaume Wenzek, Marie-Anne Lachaux, Alexis Conneau, Vishrav Chaudhary, Francisco Guzmán, Armand Joulin, and Edouard Grave. Ccnet: Extracting high quality monolingual datasets from web crawl data. **Proceedings of the 12th Language Resources and Evaluation Conference**, pp. 4003–4012, 2020.

[9] 岡崎直観, 辻井潤一. 高速な類似文字列検索アルゴリズム. 情報処理学会創立 50 周年記念全国大会, pp. 1C–1, 2010.

[10] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. Enriching word vectors with subword information. **Transactions of the Association for Computational Linguistics 2017**, pp. 135–146, 2017.

[11] 澤柳翔太, 小川泰弘, 外山勝彦. 個人向けテキスト難易度測定システムの評価. 第 20 回テキストアナリティクスシンポジウム, Vol. 123, No. 176, pp. 24–29, 2023.

[12] Wei Xu, Courtney Napoles, Ellie Pavlick, Quanze Chen, and Chris Callison-Burch. Optimizing statistical machine translation for text simplification. **Transactions of the Association for Computational Linguistics**, pp. 401–415, 2016.

A　疑似コーパスの前処理・後処理

3.1 節で述べたように, 疑似平易化コーパス構築においては, 前処理および後処理を実施した. 本節では, その処理の内容を説明する.

A.1　前処理

本研究で単言語コーパスとして用いた CC100 のサンプルは, 文単位に整理されていなかったため, 句点（”。”, ”. ”, ”!”, ”?”）を区切り文字として使用し, データを文単位に分割した.

また, 分割後に文字数が 10 文字未満, または 80 文字を超える文を削除し, 英数字の表記揺れを防ぐため, 英数字をすべて半角小文字に統一した.

A.2　後処理

構築した疑似難易度制御コーパスには, 不適切な表現や冗長なデータが残っている可能性があるため, 以下の後処理を施した.

重複削除
同一または順序が入れ替わっただけの文ペアを削除した.

連続する記号の削除
同じ記号が連続する場合, 1 つのみを残して他を削除した.

URL の削除
文中に含まれる URL（”http〜”）を削除した.

絵文字の削除
文中に含まれる絵文字を削除した.

単一語種が極端に多い文の削除
アルファベットや漢字, 記号など同一の文字種が文の 9 割を占める文を削除した.

算用数字の置換
数字部分が異なる文ペアを除外するため, 算用数字を「0」に置換した. 漢数字も置換されることが望ましいが, 慣用句や四字熟語などに用いられる可能性もあるため, 置換の対象外とした.

編集距離フィルタリング
上記の処理を実施した後, 編集距離が 0.2 未満の文ペアを除外した.

B　T23, MATCHA のデータ抽出法

本節では, 4 節の実験において比較対象, および評価データとして用いた T23, MATCHA のデータ抽出方法について説明する.

T23 についてはコーパス中のデータをそのまま使用し, 全 34,300 ペアから, 29,300 ペアを fine-tuning 用, 残り 5,000 ペアを評価用とした.

MATCHA については, 各サンプルに「テキスト対の抽出元（記事本文 or タイトル）」, 「意味内容の一致度合い（完全一致 or 部分一致）」, 「文数の対応（N 文対 M 文）」のラベルが付与されている. 本研究は文単位の平易化コーパス構築が目的のため, 全 16,000 ペアのうち, 「抽出元が記事本文」, 「意味内容が完全一致」, 「1 文対 1 文」の条件を満たす 7,376 ペアを抽出した. その後, T23 と同様の比率になるように, 6,301 ペアを fine-tuning 用, 残り 1,075 ペアを評価用として用いた.

C　モデルの出力例

本節では, 4 節で構築したモデルの出力例を示す. T23 評価データの入力・参照文と各モデルの出力例を表 4 に示す. 出力例では, Pseudo を除き, 「不在」が「いない」と平易に言い換えられている. 一方で, Pseudo では,「彼女」を「俺」に誤って言い換えている. これらの出力について, 学習元である疑似コーパスのデータを確認したところ, 不必要な削除や言い換えを含む文ペアが複数確認されたため, それらを学習していることが原因であると考えられる.

表 4　モデルの出力例（正しい変換を黄色, 誤った変換を赤字で表す）

入力文	問い合わせてみて, 彼女は不在だとわかった.
参照文	聞いてみて, 彼女は いない とわかった.
Pseudo	問い合わせてみて, 俺は不在だとわかった.
Pseudo w/DiffSim	調べてみたら, 彼女は いない とわかった.
T23	問い合わせてみて, 彼女は いない とわかった.
MATCHA	問い合わせてみて, 彼女は いないこと がわか りました.