chiVe 2.0：SudachiとNWJCを用いた実用的な日本語単語ベクトルの実現に向けて

河村 宗一郎[†1,2]　久本 空海[1]　真鍋 陽俊[1]　高岡 一馬[1]
内田 佳孝[1]　岡 照晃[3]　浅原 正幸[3]

[1]株式会社ワークスアプリケーションズ　[2]香川大学大学院工学研究科
[3]人間文化研究機構 国立国語研究所

[†]kawamura_soi@worksap.co.jp

1 はじめに

単語ベクトルは分散表現とも呼ばれ，ニューラルネットワークを用いる自然言語処理において重要な役割を果たすリソースとして，学術研究だけでなく実プロダクトにおいても利用され始めている．

我々は，異なる複数の単語分割粒度を持つ日本語単語ベクトル[1]を作成し，Sudachi Vector（chiVe：チャイブ）として公開[*1]している．chiVeは，国語研日本語ウェブコーパス（NWJC）[2]を形態素解析器Sudachi[3][*2]によって複数粒度で形態素解析を行って作られている．異なり語364万語という日本語においては類を見ない大規模な単語ベクトルである．

Sudachiは，株式会社ワークスアプリケーションズが開発・提供する形態素解析器および辞書である．

Sudachiは分割モードに応じて，短単位・中単位・長単位といった3種類の解析結果を得ることができる．例えば，「選挙管理委員会に立候補する」という入力文に対して，各分割モードで以下のような分かち書き結果を得る．

- 短単位:　選挙/管理/委員/会/に/立/候補/する
- 中単位:　選挙/管理/委員会/に/立候補/する
- 長単位:　選挙管理委員会/に/立候補/する

この分割において各粒度ごとに個別に解析を行う必要はない．Sudachiの解析辞書内には，各エントリがどの分割単位であるか，長・中単位として登録される場合には内部的にどのような分割を持つか，といった情報が付与されている．そのため，長単位の解析結果からトップダウンに分割粒度を細かくしていくことができ，個別に解析を実施する際のように，各単位間で分割境界の不整合が起きることはない．例えば，表1に示す通り，「選挙管理委員会」は長単位として登録されているが，付加情報として，中・短単位での内部的な分割情報も辞書内に保持されている．

また，Sudachiには表記の正規化機能が用意されており，送り違い，字種，異体字，誤用，縮約について，表記統制を行うことができる．

本稿では，Sudachi短単位をA単位，中単位をB単位，長単位をC単位と表す．また，B単位とC単位を合わせて「長い単位」と表し，A単位を「短い単位」と表す．

本稿で作成するベクトルの学習用コーパスとして，国語研日本語ウェブコーパス（NWJC）[2]を使用した．NWJCはウェブ上のテキストをソースとして大規模な収集を行ったものであり，約1億のウェブページのテキストを含んでいる．テキストはnwc-toolkit[*3]によって日本語文抽出とテキスト正規化，および重複文の削除を前処理として実施した．

NWJCは，国語研短単位で分割されたコーパスであるが，chiVeは，NWJCをSudachiのA単位/B単位/C単位で再度形態素解析処理することで，複数粒度の単語ベクトルとして提供している．

1.1 現在のchiVeの問題点

現在のchiVeは，語彙数が多いことに伴い，バイナリのサイズも4GB超と大きく，実プロダクト環境のように計算機リソースが限られる環境において扱いにくい．

そこで本研究では，実プロダクト環境において実用に耐えうる日本語単語ベクトルについて検討する．Sudachiの複数粒度分割を活用し，短い単位の単語のベクトルのみを保持し，これを利用してより長い単位の単語のベクトルを合成することで，大規模語彙という利点を損なわないサイズ低減を目指す．

本稿では便宜上，公開済みのchiVeを「chiVe 1.0」，本稿で説明する新手法によるchiVeを「chiVe 2.0」と呼ぶ．

表1 Sudachi登録語が保持する構成語情報

見出し	Sudachi単位	短単位分割情報	中単位分割情報
会	短単位	N/A	N/A
委員	短単位	N/A	N/A
委員会	中単位	委員/会	N/A
選挙管理委員会	長単位	選挙/管理/委員/会	選挙/管理/委員会

[*1] https://github.com/WorksApplications/chiVe
[*2] https://github.com/WorksApplications/Sudachi
[*3] https://code.google.com/archive/p/nwc-toolkit

2 関連研究

2.1 日本語単語ベクトル

2020年1月現在，chiVe 1.0[1]の他に公開されている日本語単語ベクトルとして，nwjc2vec（国語研）[4]，朝日新聞単語ベクトル（朝日新聞）[5]，HR領域向け単語ベクトル（ビズリーチ）[*4]，hottoSNS-w2v（ホットリンク）[6]がある．chiVe 1.0と，これらの比較表を表2に示す．ここで，サイズとはバイナリファイルのサイズである．

表2 一般公開されている日本語単語ベクトルの比較

	サイズ(MB)	語彙数(万)	コーパス
chiVe 1.0	4,171	364	NWJC
nwjc2vec	2,700	155	NWJC
朝日新聞単語ベクトル	907	75	朝日新聞
HR領域向け単語ベクトル	69	17	スタンバイ
hottoSNS-w2v	1,714	206	SNS

表2のうち，朝日新聞単語ベクトルとhottoSNS-w2vは，研究目的での利用が明記されている．

chiVe 1.0はApache 2.0ライセンスであり，また語彙数は類を見ない数であるが，その分サイズが大きい．

2.2 単語ベクトルの圧縮

ニューラルネットワークを用いる自然言語処理において，しばしばモデルの大きさが問題となる．朱らによれば，ベクトルの圧縮手法は4つに大別できる[7][8]．

1) 低精度計算・量子化
2) 次元削減
3) 語彙数削減
4) コーディング

朱らはこのうち，4)コーディングに着目し，深層コード学習による単語ベクトルの圧縮を提案[7][8]した．K個の基底ベクトルをM個組み合わせたK^M通りで元のベクトルを表せるように学習し，単語ベクトルを圧縮する．

我々は朱らの手法によってchiVe 1.0の圧縮を試みた．評価セットとしてJWSAN-1400[9]を用い，単語類似度と関連度について評価を行った．元の精度を損なわないパラメータを調査，圧縮したところ，M=192/K=64のとき，約2GBまで圧縮することができた．chiVe 1.0との比較を表3に示す．ここで，サイズとはバイナリファイルのサイズであり，圧縮の際のエポック数は64である．

表3 chiVe 1.0の圧縮結果

	M	K	サイズ(MB)	類似度	関連度
chiVe 1.0			4,171MB	54.06	66.53
Shu2018 [7]	192	64	1,959MB	53.36	66.35
	16	32	212MB	35.36	46.66
	32	16	327MB	32.54	43.64
	64	8	507MB	36.24	48.08

朱らは感情分析と機械翻訳を評価タスクとし，感情分析においては，M=16，K=32のとき，ベースラインの精度を損なわずに98.4%のモデル圧縮ができることを報告しているが，単語類似度や関連度に関する結果は報告していない．我々は，朱らが感情分析タスクの評価で使用していた，M=16/K=32，M=32/K=16，M=64/K=8の3つのパラメータでも圧縮を行い，単語類似度，関連度について評価した．結果はいずれも元の精度を大きく下回る結果となり，ベクトルの性質が大きく変わっていることを示している．

2.3 構成要素からの単語ベクトルの合成

朱らの手法によって，容量の削減という目的は達成することができる．しかし，圧縮された単語ベクトルは，使用するたびに復号する必要がある．また，内部の構成情報をうまく活用できておらず，元のベクトルの性質を保持していない．

そこで我々は，3)語彙数削減に着目した．単に語彙数を削減すれば容量は小さくなるが，chiVe 1.0の大規模語彙という利点が失われてしまう．そこで，長い単位の単語を削減し，短い単位を使って長い単位のベクトルを合成することを考える．

文字やサブワードから単語ベクトルを、また単語から文のベクトルを作成するアプローチは、たびたび提案されている。最もシンプルなものは、構成要素のベクトルを平均するものである．

Pinterらは，文字ベクトルから学習済みの単語ベクトルに近似するMIMICKを提案[10]した．この手法では，ある単語wの文字ベクトルvをBi-LSTMでエンコードし，多層パーセプトロンによりwのベクトルuを生成し，uをword2vecやfastTextなどで学習済みのwのベクトルに近似するようモデルおよび文字ベクトルを学習する．

ある単語のベクトルが既知語として存在すればそれを利用すればよいが，未知語の場合は対応するベクトルを作ることができない．MIMICKを使うことで，未知語の場合においてもその単語の文字からその未知語に近いベクトルを生成できる．

また，未知語に対するアプローチとして，文字単位ではなく，既知語に分割しそのベクトルを利用，加えて表層の近い既知語を利用する手法も提案[11]されている．

3 提案手法

本研究では，Sudachiの短い単位から長い単位の単語ベクトルを合成することについて検討する．

短い単位のベクトルのみを保持し，これを用いて長い単位のベクトルを合成する．長い単位のベクトルを保持しないため，その分のデータ量を削減できる．また，元となる短い単位のベクトルは，追加の計算をせずそのまま使用できる．

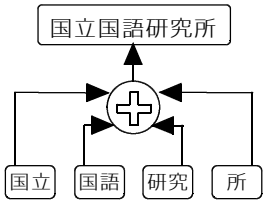

図1 長い単位の語のベクトルをより短い単位の語のベクトルから合成する機構

[*4] https://www.bizreach.co.jp/technology/research/word2vec/

短い単位から長い単位のベクトルを作るもっとも単純な方法として，各ベクトルの加算平均が考えられる．本稿では，A単位のベクトルの平均でB単位・C単位を表現する．

3.1 短い単位からの長い単位のベクトルを合成

単語ベクトルのB単位・C単位語を削除し，A単位の平均によって代替する．ここで，単語ベクトルは品詞を保持しないため，B単位・C単位を細分割する際，複数通りのA単位分割が考えられる単語が存在する．この問題に対応するため，コーパス内のB単位・C単位語についてSudachi辞書のエントリごとに出現頻度を調べ，その頻度を用いて加重平均することとした．また，最低出現回数の制約により，B単位・C単位語を細分割する際，対応するA単位語のベクトルが存在しない場合がある．B単位・C単位語を構成するA単位語のベクトルが1つでも存在しない場合は，当該のB単位・C単位語の削除行わない．

さらに，chiVe 1.0は，NWJCの形態素解析結果の表層形をそのまま使用して作られているが，本稿で検討するchiVe 2.0では，語彙の正規化表記を利用してベクトルを作成する．正規化により，単に語彙数が減るだけでなく，活用や表記揺れが1通りに統制され，ベクトル合成の際に元の概念を表現する上で都合が良いと考えられる．

4 評価と考察

chiVe 1.0は，単語の最低出現回数を5回としてSkip-gram Negative Sampling[12]で学習を行っているが，ここでは実験用に，同じ条件で最低出現回数を90回とした単語ベクトルを用意した．これを元ベクトルと呼ぶ．B単位，C単位を削除する代わりに，A単位の平均でベクトルを置き換えたベクトルを作成し，これを合成ベクトルと呼ぶ．

4.1 B単位・C単位の削減による圧縮の効果

ここで，元ベクトルの単語は48万語，このうちA単位語は22.8万語，B単位・C単位語は3.7万語で，残りの21.6万語は未知語であった．chiVe 1.0では，雑多な表現まで広く取り込まれているSudachi Full辞書を使用しているが，実験では細かな調整が施されたSudachi Core辞書を利用したため，未知語が多くなっている．全体に対するBC単位の割合は7.7%，ABC単位に対するBC単位の割合は13.9%であり，これらの語彙が削減される．

4.2 単語類似度・関連度による評価

日本語類似度・関連度データセットJWSAN-1400[9]を利用し，合成されたベクトルを評価する．ここで，JWSAN-1400は，Sudachiの正規化表記に修正する．結果を表4に示す．

表4 元ベクトルと合成ベクトルの類似度・関連度の比較

	類似度	関連度
元ベクトル	52.41	63.69
合成ベクトル	50.96	61.67

JWSAN-1400にはB単位・C単位語が存在しないため，合成ベクトル自体の評価は行えない．しかし，結果が数ポイント変化している．これは，合成されたベクトルが影響して近傍の順位が変わるからである．

4.3 文書分類による評価

下流タスクとして文書分類タスクによる評価を行う．評価データセットとして，livedoorニュースコーパス[*5]を使用する．livedoorニュースコーパスは，全7,367文書，9クラスから成る．

各文書に対してC単位で形態素解析を行い，正規化表記された名詞のベクトルの平均をとり，これを文書の特徴量ベクトルとする．この特徴量に基づき，一対多のロジスティック回帰による分類器を構築し，10分割交差検証で評価を行う．結果を表5に示す．

表5 livedoorコーパスによる文書分類の予測精度

	平均	分散
元ベクトル	0.8478	8.148×10^{-4}
合成ベクトル	0.8442	8.649×10^{-4}

元ベクトル，合成ベクトルを比較すると，ほぼ同じ精度を保っていることが分かる．この結果からは，長い単位のベクトルを平均によって代替可能と言えるが，真鍋らの実験[1]によれば，C単位の使用の有無で文書分類の結果に差がないことが報告されており，文書分類によって十分な評価ができるとはいえない．

4.4 定性的評価

文書分類は，文書中の単語の出現からクラスを予測するタスクであり，ベクトルの性質を評価することはできない．そこで，元ベクトルと合成ベクトルに対し，C単位語を入力して，コサイン類似度によって近傍を求め，上位10語を比較する定性評価を行う．

表6に「阿波踊り」に対する近傍の上位10件を示した．

表6 元ベクトルと合成ベクトルの近傍単語の違い

順位	元ベクトル		合成ベクトル	
	単語	類似度	単語	類似度
1	阿波	0.7479	阿波	0.8620
2	よさこい祭り	0.6914	踊り	0.8463
3	よさこい	0.6673	徳島阿波おどり空港	0.8452
4	盆踊り	0.6602	花笠踊り	0.7703
5	踊り	0.6279	阿波市	0.7518
6	流し踊り	0.6168	踊り念仏	0.7229
7	お囃子	0.6052	阿波銀	0.6851
8	ねぶた祭り	0.5907	田植踊	0.6733
9	囃子	0.5786	阿波弁	0.6373
10	そうらん	0.5684	阿波銀行	0.6343

「阿波踊り」は，A単位では「阿波/踊り」と分割され，合成ベクトルでは2語のベクトルの平均で表現される．元ベクトルでは，阿波踊りが何らかの祭りであることを学習しているのに対し，合成ベクトルでは語を構成する2語に強く影響されてしまうことがわかる．

このように，A単位のベクトルの単純な平均では，元の長い単位のベクトルの特徴を再現できないことがわかる．

[*5] https://www.rondhuit.com/download.html

5 今後の検討

A単位の平均は長い単位のベクトルの合成には不十分である．そこで，元のB単位・C単位の単語ベクトルを近似するようにA単位を合成する方法を考える．

5.1 よりよい合成方法の検討

2.3節で説明したMIMICKをベースとする手法が考えられる．MIMICKでは，文字を合成要素としているが，chiVe 2.0ではSudachi辞書[3]のA分割単位を合成要素とし，図2のようにB単位語およびC単位語のベクトルをA単位の単語ベクトルから合成する仕組みを作る．

Sudachi辞書の分割情報を用い，A単位語から合成することでB単位，C単位を表現する．単語ベクトルにはA単位の語彙のみを保持することで，サイズの縮小に貢献する．

2.2節で説明した朱らの圧縮手法とは異なり，A単位の単語ベクトルとしてそのまま使用できる．

MIMICKでは，合成要素となる文字ベクトルをモデルと同時に学習するが，我々の手法では，合成要素となるA単位は既に学習されているものを利用し，変更しない．

「 国立 / 国語 / 研究 / 所 」のような複合名詞を構成する各A単位語は，その複合名詞内の他のA単位語を緩やかに修飾していると考えられ，各A単位語を合成し，元のC単位語の概念に近似できると予想される．

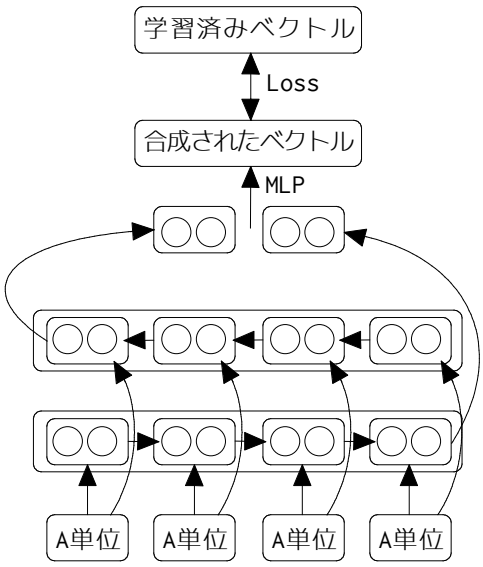

図2 Bi-LSTMと多層パーセプトロンによって
A単位から長い単位のベクトルを合成する機構

さらに，Sudachi辞書に存在しないC単位語を入力した場合にも，既知のA単位を利用して表現することができる．「徳島文理大学」「徳島工業短期大学」といった，「徳島○○大学」という用例や，「東京理科大学」「岡山理科大学」といった「○○理科大学」という用例が学習されているとき，「徳島理科大学」という未知の大学ができた場合にも，語を構成する既知語のA単位ベクトルの合成によって概念を表現できると期待される．ただし，対象とする語の範囲の特定は必要である．

5.2 長い単位のベクトルの評価手法の検討

既存の単語類似度・関連度データセットは，A単位のみが含まれている．長い単位のベクトルを適切に評価するためには，B単位・C単位語を含むテストセットが必要である．

また，文書分類，機械翻訳などの下流タスクによる評価では，長い単位の単語の出現頻度が高いとは言えず，結果に大きな差異が生じないと考えられる．

ほかにも，ベクトルの質を単語類似度や実タスクで評価するのではなく，元の学習済みベクトルと合成されたベクトルがどれだけ似ているかを評価するアプローチが考えられる．例えば，合成されたベクトルと元のベクトルの近傍を比較し，差異を定量的に確認する方法が考えられる．

6 おわりに

本稿では，日本語単語ベクトルchiVeのサイズを低減し，より実用的なものとする手法について検討した．

具体的には，Sudachiの短い単位のベクトルのみを保持し，長い単位のベクトルを合成するという手法を提案した．

A単位語のベクトルの平均では，元のベクトルを表現することはできなかった．今後は，学習済みベクトルに近似する手法について検討する必要がある．

さらに，学習済みベクトルを模倣する手法は，単語ベクトルのサイズ低減には有効に作用するが，既存の学習済みベクトルを超えるよりよいものは生成できないと考えられる．最終的には，学習済みベクトルの模倣ではなく，ABC単位を考慮したベクトル学習を行う方法を検討する．

参考文献

[1] 真鍋 陽俊，岡 照晃，海川 祥毅，高岡 一馬，内田 佳孝，浅原 正幸：“複数粒度の分割結果に基づく日本語単語分散表現”，言語処理学会第25回年次大会発表論文集，P8-5，2019.
[2] M. Asahara et al.：“Archiving and Analysing Techniques of the Ultra-large-scale Web-based Corpus Project of NINJAL, Japan”，Alexandria，Vol 26，No.1-2，pp.129-148，2014.
[3] K. Takaoka et al.：“Sudachi: a Japanese Tokenizer for Business”，Proceedings of the Eleventh International Conference on Language Resources and Evaluation，2018.
[4] M. Asahara et al.：“NWJC2Vec: Word embedding dataset from 'NINJAL Web Japanese Corpus'”，Terminology: International Journal of Theoretical and Applied Issues in Specialized Communication，Vol. 24，No. 2，pp.7-25，2018.
[5] 田口雄哉，田森秀明，人見雄太，西鳥羽二郎，菊田洸：“同義語を考慮した日本語単語分散表現の学習”，情報処理学会第233回自然言語処理研究会，Vol.2017-NL-233，No.17，pp.1-5，2017.
[6] 松野 省吾，水木 栄，榊 剛史：“日本語大規模SNS+Webコーパスによる単語分散表現のモデル構築”，人工知能学会全国大会論文集，2019.
[7] 朱中元，中山英樹：“深層コード学習による単語分散表現の圧縮”，言語処理学会第24回年次大会発表論文集，C6-1，2018.
[8] R.Shu et al.：“Compressing Word Embeddings via Deep Compositional Code Learning”，International Conference on Learning Representations，2018
[9] 猪原 敬介，内海 彰：“日本語類似度・関連度データセットの作成”，言語処理学会第24回年次大会発表論文集，P10-6，2018.
[10] Y. Pinter et al.：“Mimicking Word Embeddings using Subword RNNs”，Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing，pp.102-112，2017.
[11] 福田 展和，吉永 直樹，喜連川 優：“複数のサブワード分割と表層類似語を用いた未知語の単語分散表現の生成”，
[12] T. Mikolov et al.：“Efficient Estimation of Word Representations in Vector Space.”，CoRR abs/1301.3781，2013.