単語分割を経由しない単語埋め込み

押切 孝将　　下平 英寿
大阪大学大学院 基礎工学研究科
理化学研究所 革新知能統合研究センター
{oshikiri, shimo}@sigmath.es.osaka-u.ac.jp

1 はじめに

word2vec [1] に代表される既存の単語埋め込みアルゴリズムの多くは，文字列ではなく，単語や複合語などの列を入力として受け取る．そのため，前処理の段階で，処理したいテキストコーパス (通常これは文字列の形式で与えられる) を適当な単位に分割しておくことが必須となっている．英語のような分かち書きされる言語では，複合語を無視して空白を単語境界とすることで分割をある程度達成できる．一方で，日本語や中国語のような分かち書きされない言語では，単語分割器や形態素解析器を用いて文字列を単語単位に分割しておく必要がある．

単語分割の目的でよく使われている形態素解析は，日本語の言語処理における重要な基礎技術であるが，難しく比較的コストが高い．よく用いられている教師あり形態素解析を適切に行うためには，処理したいコーパスを十分に網羅できる単語辞書を用意する必要がある．しかし，標準的な単語辞書では対応できない語 (例えば，特殊なドメインで使われる単語や表記ゆれ，砕けた表現，顔文字，新語など) を大量に含むコーパスを扱う場合，このコーパスを十分に網羅できる単語辞書を作成するコストは非常に高い．また，仮にコーパスを十分に網羅できる単語辞書を作成できても，形態素解析の部分での誤りを避けることは難しい．

そこで本論文では，単語分割や形態素解析の処理を経由しない新しい単語埋め込み手法 sembei (segmentation-free word embeddings) を提案する．sembei では，頻出する文字 n-gram のリストを単語辞書の代わりに用いてコーパスから文字 n-gram ラティスを構成し，このラティスの上での文字 n-gram の共起頻度を使って分散表現を構成する．そして，後処理によって文字 n-gram の分散表現に基づいて単語の分散表現を構成する．

日本語版 Wikipedia コーパスを使った評価実験では，提案手法は従来手法 (MeCab を使って単語分割したコーパスと word2vec を用いる手法) に比べスコアでは劣るものの，MeCab が誤って分割してしまう単語を拾えていることが確認できた．また，Twitter から収集したテキストを用いた実験 (図 1) では，顔文字や砕けた表現，新語などの標準的な単語辞書では対応できない語に対して，提案手法が特に有効であることを確認した．なお，提案手法の実装は GitHub で公開されている (https://github.com/shimo-lab/sembei)．

Query	Neighbors
💧	😴 😴 😳 💦 😊 💦💦 😳 😱 😔 😊😊
＼(^^)／	(≧▽≦) (*≧∀≦*) ٩(ˊᗜˋ*)و (oˊ∀ˋo) (*ˊ▽ˋ*) (๑ˊ◡ˋ๑) (❁ˊωˋ❁) ヽ(*ˊ∀ˋ)ノ ＼(^-^)／ o(^o^)o
(´;ω;`)	(;∀;) (´;ω; ` ) (;_;) (;ω;) (T_T) (´;ω;`) (泣) (T＾T) (T-T) ヽ(;▽;)ノ
シン・ゴジラ	この世界の片隅に 四月は君の嘘 風夏 君の名は。 IQ246 ちはやふる 美女と野獣 北斗の拳 ねこあつめ サザエさん
サンキュー	さんきゅー さんくす あざす あざっす ありがとさん あざーす すごーい あざます おおきに サンキュー!
○ね	氏ね くたばれ 滅べ 散れ 消えろ 去れ 爆発しろ 置いてけ 連れてけ 死ねよ
でござる	でやんす でごわす でござるよ っス であります でしゅ でござる。 でござる… でございます ですぞ

図 1: Twitter コーパスと提案手法を使って構成した分散表現における最近傍検索の結果．それぞれのクエリに対して，コサイン類似度の意味で最も近い 10 個の文字 n-gram を列挙した．詳細は第 4 章に記載した．

2 Eigenwords (OSCCA)

本節では，本研究で利用する既存の単語埋め込み手法 eigenwords (OSCCA) を導入する．Eigenwords は，正準相関分析に基づく単語埋め込みの手法群である [2]．One Step CCA (OSCCA) は，eigenwords の各種手法の中でも最も単純な手法で，単語とその文脈[1]

[1] ここでは，ある単語の周辺に出現する単語のことを指す

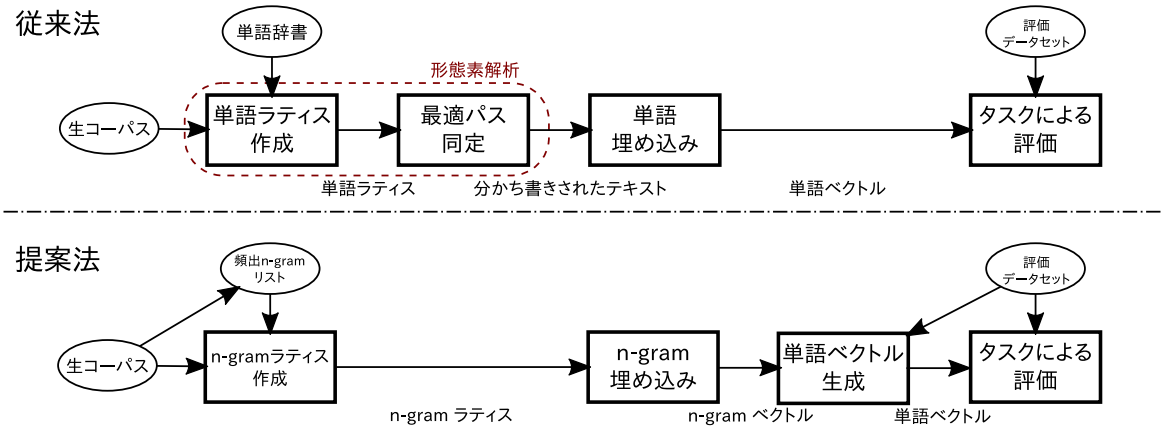

図 2: 従来の単語埋め込みのプロセスと提案手法のプロセスの比較

の共起行列 (単語文脈行列) の特異値分解を使って分散表現を構成する．

ここでは文脈窓 (context window) の幅を 1 に固定して説明する．T 個の単語トークンからなるコーパス $(t_1, t_2, \ldots, t_T)$ を考える．各 t_i は V 個の単語タイプからなる集合 $\mathcal{V} = \{v_1, v_2, \ldots, v_V\}$ に含まれるとする．$v, v' \in \mathcal{V}$ に対して $\#(v)$ はコーパス $(t_1, t_2, \ldots, t_T)$ の中で単語 v が出現する頻度を表し，$\#(v, v')$ はコーパス中でこの 2 つの単語が v, v' の順で連続して出現する頻度を表すとする．このとき，文字 n-gram とその左隣に出現する文字 n-gram の共起行列 $\mathbf{C}_L$，文字 n-gram とその右隣に出現する文字 n-gram の共起行列 $\mathbf{C}_R$，それらを列方向に結合した行列 $\mathbf{C}$ をそれぞれ，

$$\mathbf{C}_L := \left(\frac{\#(v_j, v_i)}{\sqrt{\#(v_i)}\sqrt{\sum_k \#(v_j, v_k)}} \right)_{i,j} \in \mathbb{R}^{V \times V} \quad (1)$$

$$\mathbf{C}_R := \left(\frac{\#(v_i, v_j)}{\sqrt{\#(v_i)}\sqrt{\sum_k \#(v_k, v_j)}} \right)_{i,j} \in \mathbb{R}^{V \times V} \quad (2)$$

$$\mathbf{C} := [\mathbf{C}_L,\ \mathbf{C}_R] \quad (3)$$

で定義し，さらに $\mathbf{C}$ の要素をすべて $1/2$ 乗した行列を $\sqrt{\mathbf{C}}$ で表す．OSCCA では単語の分散表現を $\mathbf{G}_1^{-1/2}[\boldsymbol{u}_1, \cdots, \boldsymbol{u}_K]$ で構成する．ただし，$\mathbf{G}_1$ は $\#(v_1), \#(v_2), \ldots, \#(v_V)$ を対角成分に並べた V 次対角行列を表し，$\boldsymbol{u}_1, \boldsymbol{u}_2, \ldots, \boldsymbol{u}_K$ は上位 K 個の特異値に対応する $\sqrt{\mathbf{C}}$ の左特異ベクトルを表す．

3 提案手法

従来の単語埋め込み手法はコーパスから単語の分散表現を構成するが，提案手法 sembei は文字 n-gram の分散表現を構成する．提案手法と既存のアプローチ (形態素解析器と既存の単語埋め込み手法を用いるアプローチ) の比較を図 2 に示す．

既存のアプローチでは，まず生コーパスに対して形態素解析を適用する．形態素解析では，単語辞書を用いてコーパス中の文章を，各ノードとエッジにコストが付いた単語ラティスに変換する．そして，単語ラティス上での最適パスを求めることで，単語分割，品詞タグ付け，原形の復元を同時に行う．このようにして得られた分割済みテキスト (単語の列) を，単語埋め込みアルゴリズムに入力し，出力として単語ベクトルを得る．

提案手法では，既存のアプローチのようにコーパスを単語の列に直してから単語埋め込みを実行するのではなく，コーパスから直接文字 n-gram ベースのラティスを構成した後，そのラティスの上での文字 n-gram の共起情報を使って埋め込みを構成する．これは，既存のアプローチにおいてラティスの上の最適パスにおける単語の共起情報のみを考えていた部分を，最適パス上に限定しないで，あらゆるパスの上での共起情報を考慮に入れることに相当する．また，単語ラティスの構成の際に，既存のアプローチでは単語辞書を使っているが，提案手法ではその代わりにコーパス中の頻出文字 n-gram のリストを使う．提案手法では，以上の工夫によって単語辞書を使った前処理を省いている．

以下では，提案手法の詳細を説明する．既存の単語埋め込み手法における語彙に相当する，V 個の文字 n-gram からなる集合を $\mathcal{V} = \{v_1, v_2, \ldots, v_V\}$ とする．本論文の実験では V 個の頻出文字 n-gram $(n = 1, 2, \ldots, 8)$ を抽出し，$\mathcal{V}$ とした．$v, v' \in \mathcal{V}$ に対して，$\#(v)$ はコーパス中で v が出現する頻度を表し，$\#(v, v')$ はコーパス中でこの 2 つの文字 n-gram が v, v' の順で連続して出現する頻度を表すとする．そして，今定義した $\#(\cdot)$, $\#(\cdot, \cdot)$ を用いて，OSCCA の

ときと同様に $v_1, v_2, \ldots, v_V$ の分散表現を構成する．

分散表現は $\mathcal{V}$ に含まれるすべての文字 n-gram に対して与えられているため，必要であれば，文字 n-gram の分散表現から単語の分散表現を構成する．ただし，本論文で行った実験では，このステップを省略した．

なお，本研究では共起行列の構成や重み付けなどにおいて OSCCA をベースにして議論を進めたが，他の単語文脈行列を利用する単語埋め込み手法をベースに同様の拡張を考えることもできる．

4 実験

この節では，提案手法 sembei の有効性を示すために，形態素解析器と従来の単語埋め込みを使うアプローチによって得られた分散表現と sembei によって得られた分散表現の性能比較を行った．

4.1 実験の設定

分散表現は，日本語版 Wikipedia のダンプデータ [2] から HTML タグなどを取り除いたテキストコーパス (Wikipedia コーパス) を用いて構成した．分散表現の次元は 200 で統一した．

sembei で用いる頻出文字 n-gram ($n = 1, 2, \ldots, 8$) の抽出には，大規模なストリームデータから頻出アイテムを抽出する手法である，誤り許容カウント法 (lossy counting algorithm) [3] を用いた．抽出する文字 n-gram の数は，1,150,000 個とした．また，行列の特異値分解の計算には，Dhillon ら [2] と同様に，大規模行列の特異値分解を効率的に実行することができる乱択化アルゴリズム [4] を用いた．

ベースライン手法としては，標準的な形態素解析器 MeCab[3] v0.996 (単語辞書は mecab-ipadic v2.7.0) を使って分割の前処理を行った Wikipedia コーパスと word2vec [1] を用いた．以下，この手法を word2vec+MeCab と呼ぶ．語彙としては，分割済みコーパス中に 5 回以上出現する単語 958,724 語を用いた．なお，word2vec+MeCab で使用する語彙中の単語の中で sembei の語彙にも含まれている単語の割合は，約 10.5%である．

構成した単語ベクトルの評価は，日本語単語類似度データセット [5] によって行った．単語類似度データセットによる評価では，単語間の類似度をそれぞれの単語の分散表現の間のコサイン類似度によって推定する．日本語類似度データセットにおける単語の類似度と，分散表現によって推定した単語の類似度の間のスピアマン順位相関係数によって性能を評価した．

また，単語類似度データセットによる評価とは別に，Twitter コーパスを使った実験も実施した．Twitter コーパスは，2016/10/27 から 2016/11/22 にかけて Twitter Streaming API を使って収集した 17,316,968 件の日本語ツイートからなる．このコーパスと提案手法により構成した分散表現を使って，適当に選んだクエリに最も似ている 10 個の文字 n-gram を検索した．

4.2 実験結果

日本語単語類似度データセットを使った性能評価の結果を表 1 に示す．いずれの設定においても提案手法は正の相関を示しているものの，相関係数の値では word2vec+MeCab より悪いスコアになっている．

いくつか適当なクエリを選んで，コサイン類似度に基づいて最近傍検索したときの結果を表 2 に示す．表 2 の結果より，word2vec+Mecab と同様に，提案手法でも似たトピックの単語を抽出できていることがわかる．

提案手法では，陽に分割の前処理をしなかったため，語彙にそれ自体が単語にではない文字 n-gram が大量に含まれている．そのため提案手法には，word2vec+MeCab では語彙に含まれているが比較的出現頻度が低い語を拾うことができないという欠点がある．一方で，word2vec+MeCab では拾えなかった単語を拾える場合がある．例えば，今回用いた MeCab では，“C++” という文字列を “C” と “++” の 2 語に分割してしまう．そのため，word2vec+MeCab で用いた語彙には “C++” は含まれておらず，結果として “C++” の分散表現は獲得できない．一方で，提案手法では “C++” という 3-gram は語彙に入っているため分散表現が構成できており，さらに “Java” に近い単語として “C++” が得られている．

次に，Twitter コーパスを用いた最近傍探索の結果を図 1 に示す．図 1 より，提案手法は単語辞書を用いていないものの，顔文字や新語といった分割が困難な語に対しても類似語をピックアップできていることがわかる．また顔文字や絵文字については，類似した極性の顔文字や絵文字に似たベクトル表現が与えられる傾向があった．

[2] https://dumps.wikimedia.org/jawiki/latest/jawiki-latest-pages-articles.xml.bz2 (2016/10/21 時点のデータを使用した)

[3] http://taku910.github.io/mecab/

	word2vec +MeCab	sembei
adj	20.59	7.96
adv	22.76	13.93
noun	28.21	14.70
verb	28.94	24.68

表 1: 日本語単語類似度データセットを用いた単語ベクトルの性能評価の結果．数字は，スピアマン相関係数を 100 倍した値である．

	Google	Java	ゴジラ
word2vec +MeCab	グーグル	API	モスラ
	検索	Python	キングギドラ
	google	コンパイラ	ガメラ
	Chrome	++	デストロイア
	サーチ	Perl	怪獣
sembei	Amazon	Linux	ガメラ
	Goog	XML	モスラ
	グーグル	Unix	マジンガー Z
	ウィキメディア	C++	スーパーマン
	User	HTTP	キン肉マン

表 2: クエリとなる文字 n-gram とそれに最も近い 5 個の文字 n-gram．5 個の文字 n-gram は，既存手法 (word2vec+MeCab) および提案手法 (sembei) によって構成した分散表現を使って求めた．

5 関連研究

本研究では，分かち書きされない言語，特に日本語における単語埋め込みを扱った．英語に代表される分かち書きされる言語においても，動詞の活用や表記ゆれ，トークナイズによる誤りなどの影響を軽減したいというモチベーションのもとに，単語に含まれる形態素や部分文字 n-gram の情報を援用した単語埋め込みの手法がいくつか提案されている．このような研究の中でも特に本研究と関連度が高いものとしては，単語の部分文字 n-gram の情報を利用した単語埋め込み [6]，ランダムに切り出した文字 n-gram を使った単語埋め込み [7] が挙げられる．

6 おわりに

本論文では，単語辞書と単語分割を必要としない単語埋め込みの手法 sembei を提案した．提案手法を用いることで，日本語の生コーパスから単語の分散表現を直接構成することができた．単語類似度タスクにおける性能では単語辞書と形態素解析器を用いる既存のアプローチに対して劣るものの，単語辞書を用いずに Wikipedia コーパスや Twitter コーパスから，表記ゆれや砕けた表現，顔文字，新語などの分散表現を直接構成することができた．

提案手法では，単語埋め込みの際に用いる語彙の構成において単純な頻出文字 n-gram のリストを用いたが，この部分を工夫することが考えられる．例えば，語彙として用いる文字 n-gram の抽出において，配布されている単語辞書や，正規表現を用いたヒューリスティック (例えばカタカナの列は必ず最長で取るようにするなど) を活用するようなハイブリッドアプローチを考えることができる．このように，従来の形態素解析で対応できる部分では整備された言語資源や知見を活用しつつ，新語や砕けた表現といった通常の単語辞書では対応できないような語を含む文章に対応することで，提案手法の長所を活かしながら性能を向上させることが見込める．

謝辞

本研究は，JSPS 科研費 JP16H01547 および JP16H02789 の助成を受けたものです．

参考文献

[1] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in NIPS*, pp. 3111–3119, 2013.

[2] Paramveer S. Dhillon, Dean P. Foster, and Lyle H. Ungar. Eigenwords: Spectral word embeddings. *Journal of Machine Learning Research*, Vol. 16, pp. 3035–3078, 2015.

[3] Gurmeet Singh Manku and Rajeev Motwani. Approximate frequency counts over data streams. In *Proceedings of VLDB '02*, pp. 346–357. VLDB Endowment, 2002.

[4] Nathan Halko, Per-Gunnar Martinsson, and Joel A. Tropp. Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions. *SIAM review*, Vol. 53, No. 2, pp. 217–288, 2011.

[5] 堺澤勇也, 小町守. 日本語動詞･形容詞類似度データセットの構築. 言語処理学会第 22 回年次大会論文集, pp. 262–265. 言語処理学会, 2016. https://github.com/tmu-nlp/JapaneseWordSimilarityDataset.

[6] John Wieting, Mohit Bansal, Kevin Gimpel, and Karen Livescu. Charagram: Embedding words and sentences via character n-grams. In *Proceedings of EMNLP2016*, pp. 1504–1515. Association for Computational Linguistics, 2016.

[7] Hinrich Schuetze. Nonsymbolic text representation, 2016. *arXiv preprint arXiv:1610.00479v2*.