学習済み分散表現への新規単語埋め込みに関する検討

Consideration of new word embedding to learned distributed representation.

森　祥恭[1]　柴田　祐樹[1]　高間　康史[1]

Yoshiyuki Mori[1], Hiroki Shibata[1], and Yasufumi Takama[1]

[1] 首都大学東京大学院システムデザイン研究科

[1] Graduate School of System Design, Tokyo Metropolitan University

Abstract: This paper proposes a method to find vectors of new words on the embedding space without additional learning. While distributed representation is becoming an essential technique for utilizing text information, it has a problem that it cannot calculate embedding vectors of words not included in a corpus. This paper aims to generate the embedding vectors of such words by using the embedding vectors of the words belonging to the same category as the target word. This paper proposes 3 types of calculation methods, and shows the result of preliminary experiments.

1　はじめに

近年，自然言語処理の分野において単語の分散表現を獲得する手法が注目を集めている．単語の分散表現は，テキストコーパスを利用して学習することにより獲得され，コーパス内に存在する単語をベクトルで表現できる．単語の分散表現は機械翻訳[1]やテキストに書かれた感情の分析[2]など多様な用途で利用されている．単語の分散表現を獲得する代表的な手法として，Word2Vec[3]，FastText[4]等が挙げられる．

Word2Vec などの単語の分散表現の特徴として，学習した単語ベクトル間にコサイン類似度を計算することで，単語間の類似度を求めることができる点が挙げられる．また，加法構成性を持つこと，すなわち国と首都や男性と女性のような意味的な関係も単語ベクトル間に関係に反映されていることが挙げられる．しかし，コーパス内に存在しない単語の埋め込みベクトルが得られないことや，特にコーパス内での出現頻度が少ない単語の場合など，学習によって得られたベクトルの信頼性が低いなどの問題点がある[5]. このような問題に対して，コーパスの拡張や subword を用いて対処する研究も行われている[5][6].

上記とは異なるアプローチとして，本稿では，既に学習済みの単語ベクトルからコーパスに存在しない単語のベクトル表現を生成することを最終的な目標とする．学習済みの単語ベクトルを利用して，それらと意味が近い単語ベクトルを生成できれば，大規模コーパスを用いずに語彙数の増加が可能になると考える．その前段階として，生成対象の単語と同カテゴリに存在する単語の埋め込みベクトルなどを利用して，当該単語の埋め込みベクトルを求める手法を提案する. 単一のカテゴリの関係を用いて生成する方法，1 種類の意味的もしくは構文的な関係を用いて生成した単語ベクトルにカテゴリの重心ベクトルを用いて平行移動させる方法，複数のカテゴリ間の関係を用い生成する方法の 3 種類の計算方法を提案し，予備実験を行った結果を示す.

2　関連研究

2.1　分散表現

従来，単語表現として一般的に利用されてきた one-hot 表現は，扱う単語数の分だけ次元を持ち，各要素をそれぞれの単語に割り当てて単語をベクトル表現していた．これに対し，分散表現とは各単語を高次元の実数ベクトルで表現したものであり，テキストコーパスを利用することで求められる．一般に 50～300 次元で表現される．実数のベクトルで表現されることから単語間の類似度を求めることができる．また，加法構成性の性質を持つことから，ベクトル演算を行うことで単語間の意味的な関係を利用することもできる. 例えば，“Paris”-“France”+“Japan”の演算結果が“Tokyo”になるような演算を行える．

2.2　Word2Vec

Word2Vec は，2 層からなるニューラルネットワークを用いて単語の分散表現を学習する手法である．学習する方法として CBoW（Continues Bag-of-Words）モデル[7][8]と Skip-gram モデル[8][9]の二つのモデ

ルが提案されている．CBoW モデルはテキスト内の周辺にある単語を利用して対象となる単語を予測し，単語の分散表現を学習する．Skip-gram モデルは，ある単語が与えられた時に，その周辺の単語を予測し，単語の分散表現を学習する．

2.3 subword による単語ベクトル生成

単語の分散表現はテキストコーパスから獲得するため，1 節で述べた通り，コーパスに存在しない単語や出現頻度の低い単語に対し適切な単語ベクトルが求められないという問題点がある．その解決策の一つとして，単語より小さな単位である subword を用いて分散表現を獲得する手法が提案され，FastText[4]等で利用されている．この手法では，単語の文字 n-gram に対する特徴ベクトルを計算し，それらを足し合わせたものを単語のベクトルとする．例として，“where”という単語を挙げると，文字 n-gram は“wh”，“whe”，“her”，“ere”，“re”となり，それぞれのベクトルを足し合わせることで，“where”の単語ベクトルを生成する．未知語だけでなく略語にも対応できること，品詞の活用形を考慮した単語ベクトルの生成ができることなどの利点が存在する．

3 提案手法

3.1 学習データ

本稿では，Word2Vec で学習した単語の分散表現を利用して，コーパスに存在していない単語のベクトル表現を生成する．Word2Vec の学習には，英語版 Wikipedia のオリジナルのテキストデータに前処理を施した text8[1]を利用する．このデータセットの単語総数は 253,854 である．

英単語のカテゴリ間の関係タイプとして，意味的と構文的の 2 種類を利用する．単語の意味的なカテゴリには英語の概念辞書である WordNet[2]を，構文的なカテゴリに関しては，英語の形態素解析ツールである TreeTagger[3]の品詞情報を利用する．単語の意味的な関係を表 1 に，構文的な関係を表 2 に示す．

表 1. 意味的な関係

カテゴリ 1	カテゴリ 2
国	首都
国	通貨
通貨	首都
男性	女性

表 2. 構文的な関係

カテゴリ 1		カテゴリ 2	
品詞	活用形など	品詞	活用形など
名詞	単数形	名詞	複数形
動詞	単数形	動詞	複数形
動詞	単数形	動詞	過去形
動詞	単数形	動詞	過去分詞
動詞	単数形	動詞	現在分詞
形容詞	原形	形容詞	比較級
形容詞	原形	形容詞	最上級
形容詞	原形	副詞	原形
副詞	原形	副詞	比較級
副詞	原形	副詞	最上級
副詞	原形	名詞	単数形
副詞	原形	名詞	複数形

3.2 単語ベクトルの生成方法

3.1 節で分類した関係タイプを利用して単語ベクトルを生成する．1 つ目の生成手法は，単一のカテゴリの関係を用いて生成する．生成したい単語xの属するカテゴリをc_1，c_1と関係するカテゴリc_2とし，以下の手順により図 1 に示す関係にある単語対の集合Pを求める．

(1) c_2に属し，xと対応関係にある単語yを求める．
(2) c_2に属する単語のうち，y以外で式（1）（2）を満たす単語の集合Yを求める．
(3) Yに属する各単語y'について，c_1に属する単語x'を求め，単語対(x', y')の集合Pを求める．

式（1）（2）におけるd_c，d_l，d_hはパラメータである．求めた$\boldsymbol{x}'$，$\boldsymbol{y}$，Pを利用して$\hat{\boldsymbol{x}}_1$を式（3）により生成する．

$$\cos(\boldsymbol{y}, \boldsymbol{y}') \geq d_c \quad \cdots (1)$$

$$d_l \leq \sqrt{|\boldsymbol{y}' - \boldsymbol{y}|} \leq d_h \quad \cdots (2)$$

$$\hat{\boldsymbol{x}}_1 = \frac{\sum_{(x', y') \in P} (\boldsymbol{x}' - \boldsymbol{y}' + \boldsymbol{y})}{|P|} \quad \cdots (3)$$

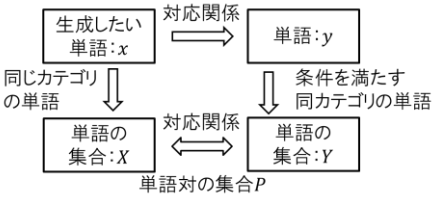

図 1. 単語間の関係図

[1] http://mattmahoney.net/dc/text8.zip -O text8.gz
[2] https://wordnet.princeton.edu/
[3] https://www.cis.uni-muenchen.de/~schmid/tools/TreeTagger/

2 つ目の生成方法は，式（3）で計算した単語ベクトルをカテゴリの重心ベクトルに基づき移動させて生成する．C_1のカテゴリの重心ベクトルを$\boldsymbol{g_1}$，C_2の重心ベクトルを$\boldsymbol{g_2}$として，生成式を式（4）に示す．

$$\hat{\boldsymbol{x}}_2 = \hat{\boldsymbol{x}}_1 + \boldsymbol{g_1} - \boldsymbol{g_2} \quad \cdots (4)$$

3 つ目の生成手法は複数の関係を用いて単語のベクトルを生成する．複数のクラスの平均を求める場合，クラスごとのデータの件数の比率を考慮するマクロ平均と，考慮しないマイクロ平均の 2 種類が存在する．この 2 種類の平均を利用してそれぞれ単語ベクトルを生成する．関係の総数をL，i番目の関係を用いて式（3）より生成したベクトルを$\hat{\boldsymbol{x}}_{1,i}$としたときのマクロ平均での生成式を式（5）に示す．

$$\hat{\boldsymbol{x}}_3 = \sum_{i=1}^{L} \frac{\hat{\boldsymbol{x}}_{1,i}}{L} \quad \cdots (5)$$

i番目の関係についての単語対集合をP_i，xに対応する単語をy_iとしたときのマイクロ平均での生成式を式（6）に示す．

$$\hat{\boldsymbol{x}}_4 = \frac{\sum_{i=1}^{L} \sum_{(x',y') \in P_i} (x' - y' + y_i)}{\sum_{i=1}^{L} |P_i|} \quad \cdots (6)$$

4 予備実験

4.1 実験概要

本稿では予備実験として，単語ベクトル既知の単語について提案手法によりベクトル表現を生成し，既知のベクトルと比較することで評価を行う．評価指標として，コサイン類似度とユークリッド距離を採用する．式（1）（2）のパラメータとして，（d_c，d_l，d_h）=（0.5，0，15）とした．

予備実験で使用する関係，単語数，単語の例を表 3 に示す．表 3 における関係の左側にある単語のカテゴリのみから単語$\boldsymbol{x}$を選んで実験を行う．単語対の数は今回使用した対の総数で，対象単語ごとにベクトル生成に利用される単語対は異なる．

4.2 実験結果

単語の分散表現として，Word2Vec を以下の条件で学習したものを使用した．

- 単語ベクトルの次元数：200
- コーパス内での出現回数が 5 回未満の単語を無視
- 予測する単語の前後 5 単語を学習に使用
- 学習モデル：CBoW

首都，動詞の単数形，副詞のそれぞれの単語ベクトルを生成し，コサイン類似度で評価した結果をそれぞれ表 5，8，11 に，ユークリッド距離で評価した結果をそれぞれ表 6，9，12 に示す．複数の関係がなく単一の関係でのみ単語ベクトルを生成した場合に「なし」としている．各単語における使用した単語対数をそれぞれ表 4，7，10 に示す．単語対は，3.2 節で説明した生成手順に従い，それぞれの$\boldsymbol{x}$に対して選ぶ．

表 3. 実験で使用する関係，単語対の数と例

関係	単語対数	単語対の例
首都-国	32	Tokyo-Japan Beijing-China
首都 -ISO code	17	Tokyo-jpy Bern-chf
単数形-現在分詞(動詞)	73	Sing-Singing Dance-Dancing
単数形-複数形（動詞）	80	Sing-Sings Kiss-Kisses
単数形-過去形（動詞）	116	Sind-Sang Shout-Shouted
単数形-過去分詞(動詞)	39	Sing-Sung Hear-Heard
副詞-形容詞	35	Interestingly-Interesting Obviously-Obvious
副詞-単数形（名詞）	27	Interestingly-Interest Importantly-Importance
副詞-複数形（名詞）	30	Interestingly-Interests Idealy-Ideals
副詞-比較級	10	Happily-happier Fairly-Fairer
副詞-最上級	4	Happily-happiest Sadly-Saddest
男性-女性	39	Man-Woman Boy-Girl
単数形-複数形（名詞）	120	Man-Men Woman-Women
形容詞-比較級	73	Big-Bigger Tall-Taller
形容詞-最上級	22	Big-Biggest Large-Largest

首都に関して，$\hat{\boldsymbol{x}}_1$，$\hat{\boldsymbol{x}}_2$は国のカテゴリのみを利用し，それ以外は ISO code のカテゴリも使用して生成した．表 5，6 より$\hat{\boldsymbol{x}}_2$は$\hat{\boldsymbol{x}}_1$より正解データとの違いが大きくなっていることがわかる．一方，$\hat{\boldsymbol{x}}_3$，$\hat{\boldsymbol{x}}_4$では両方とも類似度が高くなっている．

表 4. 首都の単語対数

x	国	ISO code
Tokyo	12	10
Paris	11	3
Cairo	8	10
Beijing	12	0

表 5. 首都のベクトルを生成した際の
正解データとのコサイン類似度

x	$\hat{\boldsymbol{x}}_1$	$\hat{\boldsymbol{x}}_2$	$\hat{\boldsymbol{x}}_3$	$\hat{\boldsymbol{x}}_4$
Tokyo	0.4679	0.3295	0.6437	0.6069
Paris	0.7014	0.6909	0.7083	0.7169
Cairo	0.6313	0.434	0.6725	0.6748
Beijing	0.5530	0.3802	なし	なし

表 6. 首都のベクトルを生成した際の
正解データとのユークリッド距離

x	$\hat{\boldsymbol{x}}_1$	$\hat{\boldsymbol{x}}_2$	$\hat{\boldsymbol{x}}_3$	$\hat{\boldsymbol{x}}_4$
Tokyo	9.4355	10.4886	5.7595	6.2138
Paris	10.2954	10.431	10.2528	9.9456
Cairo	7.6408	9.5623	4.7759	4.6508
Beijing	9.1188	10.429	なし	なし

動詞の単数形に関して，$\hat{\boldsymbol{x}}_1$，$\hat{\boldsymbol{x}}_2$は動詞の現在分詞のカテゴリのみを利用し，それ以外は複数形，過去形, 過去分詞のカテゴリも使用して生成した. “scan”の$\hat{\boldsymbol{x}}_2$およびユークリッド距離で比較した場合の“train”の$\hat{\boldsymbol{x}}_2$を除いて，$\hat{\boldsymbol{x}}_1$が最も正解データとの違いが大きくなる結果が得られた．最も正解データに近くなる生成方法は$\hat{\boldsymbol{x}}_3$か$\hat{\boldsymbol{x}}_4$のどちらかである.

表 7. 動詞の単数形の単語対数

x	現在分詞	複数形	過去形	過去分詞
Sing	5	30	24	14
Take	5	13	10	25
Train	8	32	22	22
Scan	15	5	16	16

表 8. 動詞の単数形のベクトルを生成した際の
正解データとのコサイン類似度

x	$\hat{\boldsymbol{x}}_1$	$\hat{\boldsymbol{x}}_2$	$\hat{\boldsymbol{x}}_3$	$\hat{\boldsymbol{x}}_4$
Sing	0.6334	0.6836	0.8107	0.8684
Take	0.8184	0.8433	0.9061	0.8802
Train	0.4227	0.4285	0.6738	0.8124
Scan	0.6779	0.6498	0.7503	0.7404

表 9. 動詞の単数形のベクトルを生成した際の
正解データとのユークリッド距離

x	$\hat{\boldsymbol{x}}_1$	$\hat{\boldsymbol{x}}_2$	$\hat{\boldsymbol{x}}_3$	$\hat{\boldsymbol{x}}_4$
Sing	8.6609	8.2400	4.6622	4.2194
Take	9.6607	9.0615	7.2080	8.0084
Train	14.2758	14.5287	8.2631	6.4012
Scan	4.6218	5.0638	3.2080	3.2843

副詞のベクトル$\hat{\boldsymbol{x}}_1$，$\hat{\boldsymbol{x}}_2$は形容詞のカテゴリのみを利用し，それ以外は単数形，複数形，比較級，最上級のカテゴリを使用して生成した．コサイン類似度に関しては$\hat{\boldsymbol{x}}_2$が最も正解データに近くなる結果が得られた．$\hat{\boldsymbol{x}}_3$，$\hat{\boldsymbol{x}}_4$はコサイン類似度で，正解データとの違いが$\hat{\boldsymbol{x}}_1$よりも大きくなる傾向にあることが確認された．首都や動詞の単数形に比べても，正解データとの違いは大きいため，計算に使用する関係の組み合わせなどを変更するなどして，さらに検討を進める必要があると考える.

表 10. 副詞の単語対数

x	形容詞	単数形（名詞）	複数形（名詞）	比較級	最上級
Apparently	7	4	6	0	0
Interestingly	15	8	7	0	0
Possibly	7	10	16	0	0
Happily	8	8	0	10	4

表 11. 副詞のベクトルを生成した際の
正解データとのコサイン類似度

x	$\hat{\boldsymbol{x}}_1$	$\hat{\boldsymbol{x}}_2$	$\hat{\boldsymbol{x}}_3$	$\hat{\boldsymbol{x}}_4$
Apparently	0.4818	0.5185	0.2896	0.3259
Interestingly	0.2896	0.2937	0.0526	0.1461
Possibly	0.2978	0.3110	0.3051	0.2969
Happily	0.4535	0.4664	0.4651	0.4641

表 12. 副詞のベクトルを生成した際の
正解データとのユークリッド距離

x	$\hat{\boldsymbol{x}}_1$	$\hat{\boldsymbol{x}}_2$	$\hat{\boldsymbol{x}}_3$	$\hat{\boldsymbol{x}}_4$
Apparently	8.0499	7.9195	9.2271	8.7082
Interestingly	7.4403	7.4184	6.7586	6.2766
Possibly	11.3965	11.4336	8.7087	8.4084
Happily	5.9463	6.1349	2.5973	2.6462

首都，動詞の単数形，副詞以外のカテゴリについても, 表 3 に示した全ての関係について実験を行い，それらをまとめた結果を表 13 に示す. 対象単語のカテゴリをカテゴリ 1，単一の関係に使用したものをカテゴリ 2 としている．コサイン類似度とユークリ

ッド距離の両方が$\hat{\boldsymbol{x}}_1$より正解データに近くなった場合は○，その中で最も近くなった場合を◎，どちらかのみ近くなった場合は△，両方とも正解データとの違いが$\hat{\boldsymbol{x}}_1$より大きい場合を×としている．

全体的な傾向としては，このことから，3 つ目の生成手法（$\hat{\boldsymbol{x}}_3$，$\hat{\boldsymbol{x}}_4$）が良好であるが，名詞の単数形では$\hat{\boldsymbol{x}}_2$が優れていることなどから，カテゴリによって適切なものを選ぶことで良い結果が得られる可能性が示された．

表 13. 各手法のまとめ

カテゴリ 1	カテゴリ 2	$\hat{\boldsymbol{x}}_2$	$\hat{\boldsymbol{x}}_3$	$\hat{\boldsymbol{x}}_4$
首都	国	×	◎	◎
動詞の現在形	動詞の現在分詞	△	◎	◎
副詞	形容詞	△	△	△
男性	女性	○	○	◎
名詞の単数形	名詞の複数形	◎	×	×
形容詞	比較級	△	◎	△

5 おわりに

本稿では，単語の分散表現を学習する手法である Word2Vec により得られた単語ベクトルを使用して，追加学習なしに新たな単語ベクトルを生成する手法を提案し，予備実験を行った結果を示した．単語間の関係や，カテゴリの重心ベクトルを利用して単語のベクトル表現を生成することにより，正解のデータに近いものを生成できることを示した．また，カテゴリごとに適した計算方法があることも示した．

本稿で示した実験では，コーパス内で出現頻度の比較的高い単語について生成を行ったため，今後は，未知語やコーパス内での出現頻度が低い単語のベクトル表現を生成し，実験を行っていく．また，他の生成手法や，単語ベクトルではなく，文書分類や要約などのタスクレベルでの評価方法などを検討する予定である．

参考文献

[1] K. Cho, D. Bahdanau, F. Bougares, H. Schwenk, Y. Bengio: Learning Phrase Representations Using RNN Encoder-Decoder for Statistical Machine Translation, Proc. of EMNLP, pp. 1724-1734（2014）

[2] D. Tang, F. Wei, N. Yang, M. Zhou, T. Liu, B. Qin: Learning Sentiment-Specific Word Embedding for Twitter Sentiment Classification, ACL2014, pp. 1555-1565 (2013)

[3] Google Code Archive: Word2Vec, https://code.google.com/archive/p/word2vec/ (2013) 最終アクセス 2019 年 6 月 20 日.

[4] A. Joulin, E. Grave, P. Bojanowski, T. Mikolov: Bag of Tricks for Efficient Text Classification, ACL2017, vol.2, pp. 427-431 (2017)

[5] M-H. Luong, R. Socher, C. D. Manning: Better Word Representations with Recursive Neural Networks for Morphology, CoNLL2013, pp. 104-113 (2013)

[6] J. Zhao, S. Mudgal, Y. Liang: Generalizing Word Embeddings Using Bag of Subword, EMNLP2018, pp. 601-606 (2018)

[7] Q. Le, T. Mikolov: Distributed Representations of Sentences and Documents, ICML2014, pp. 1188-1196, (2014)

[8] T. Mikolov, K. Chen, G. Corrado, J. Dean: Efficient Estimation of Word Representations in Vector Space，ICLR Workshop 2013 (2013)

[9] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, J. Dean: Distributed Representations of Words and Phrases and their Compositionality，NIPS2013，pp. 3111-3119 (2013)