Vol.2019-DBS-169 No.15
Vol.2019-IFAT-136 No.15
2019/9/10

分散表現空間間の変換行列による擬人化の比喩表現の自動生成

高橋 克郎[†1,a)]　大島 裕明[†1,b)]

概要：擬人化は頻出する比喩表現の一つである．我々は（主語＋述語）型の擬人化（例　飛行船が泳ぐ）の生成に関する手法を提案する．提案手法は人ではない主語と述語のタプル（例　（飛行船，飛ぶ））を入力とする．入力された主語の擬人化として適切な擬人化のリスト（例　飛行船が泳ぐ，飛行船が走る，飛行船が笑う）を出力する．動詞は人間の動作を表す動詞を出力対象とする．提案手法は基本的には，次の二段階の処理を行う．(1) 用例中の各主語と各述語の間の比喩の成立尤度を，用例から抽出した情報を基に算出する．(2) 述語のリストを，入力された主語に対する尤度順に出力する．さらに，生成された擬人化の用例を事例分析した．事例分析の目的は，擬人化を成立に必要な性質を詳細に調べることである．この分析により主語，述語の語意をクロネッカー積で表現することの有効性を得た．用例をウェブから収集した．収集した用例を正解データとした．システムの性能を平均逆順位 (MRR) により評価した．

キーワード：分散表現，変換行列，擬人化，比喩生成

Genereting Personification Metaphors by Transformation Matrix

1. はじめに

人間以外のものの動きを人間の動きのように捉える比喩表現として擬人化は日常会話の中でよく見られる表現である．例えば，飛行船が飛ぶ動きを「飛行船が泳ぐ」と表現したり，雲が風で流されている動きを「雲が走る」と表現する．「飛行船」や「雲」は人ではないものを指す主語である．「泳ぐ」や「走る」は人の動きを表す述語である．飛行船や雲は文脈から空のものであり，泳ぐは水中での動き，走るは地面の上での動きであることが用例の文脈から推測される．したがって，飛行船，雲，泳ぐ，走るはそれぞれの単語が単独でドメインを持っていると言える．語の周辺語から語意を各語のドメインとして表現する．この手法として，Skip-gram モデル [4] があげられる．我々は単語の語意表現を Skip-gram モデルを利用して作成された日本語エンティティベクトル[*1]の分散表現を利用した．

「A における B」から最尤の「C における D」を検索する 4 項のアナロジー問題を $(A, B) \sim (C, D)$ 問題と表記する．我々は，擬人化を (A, B)∼(C, D) 問題の枠組みで捉える．この場合は「**飛行船**における**飛ぶ**」から最尤の「**人**における**泳ぐ**」と捉えることができる．提案手法は，主語と述語からなる意味をクロネッカー積で表現した．これにより，先行研究にあるような単語個別の分散表現の分析ではなく，文を直接分析した．擬人化の生成手法として語の分散表現空間間の変換行列を求める [9] の手法を基礎としている．提案手法の性能を MRR により評価した．

2. 関連研究

[6] は語彙をニューラルネットワークを用いて，語とその文脈語の関係性から語の語意を捉えベクトル（分散表現）に変換する手法である．[4], [5] は [6] を簡略化した Skip-gram モデルを提案した．Skip-gram モデルは語 $w_i \in W$ のによる分散表現（恣意的に指定可能な p 次元の実ベクトル）で

†1 現在，兵庫県立大学大学院 応用情報科学研究科
Presently with Graduate School of Applied Informatics, University of Hyogo
a) ab18y501@ai.u-hyogo.ac.jp
b) ohshima@ai.u-hyogo.ac.jp

*1 http://www.cl.ecei.tohoku.ac.jp/ m-suzuki/jawiki_vector/

Vol.2019-DBS-169 No.15
Vol.2019-IFAT-136 No.15
2019/9/10

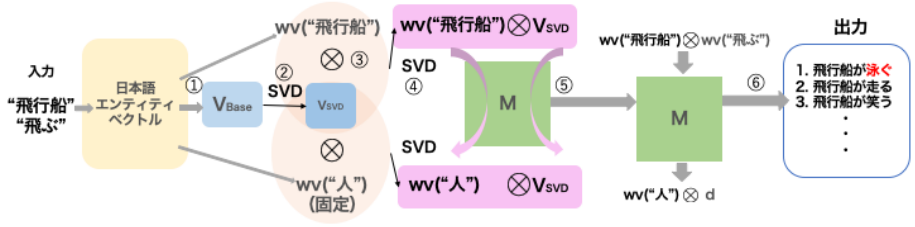

図 1 提案手法の概要

表現する．Skip-gram モデルはもともと $(A, B) \sim (C, D)$ 問題を解くために設計されたが，通常は単語の語意の指標として使用されることが多い．

擬人化は一般にメタファー（隠喩）の一種に分類されることが多い．自然言語処理において，メタファーの研究はコーパスの収集・検出・解釈・生成の 4 つのタスクに分類される．[7] が英語のコーパスで（主語＋述語）型のメタファーの解釈タスクについての研究を行なっている．WordNet[*2]の階層構造に対して選択的優先性を考慮にいれたフィルターでメタファーか字義的かの判定を行うことが基本である．この研究では，「用例再現でないメタファーで妥当なものはどれか？」などの事例分析も行なっている．

情報検索において，$(A, B) \sim (C, D)$ 問題に関する関連研究としては以下のようなものが挙げられる．[8] は関係性の類似性を尺度とする手法を提案した．彼らは，語の組 (A, B) が与えられた時の言語的な $(A, B) \sim (C, D)$ を解こうとした．この手法は，SVM による分類がもっとも精度が高かった [8]．[2] はこの問題を Web 検索エンジンのインデックスに適用した．この研究で取り組んだ $(A, B) \sim (C, D)$ は，例えば $(iPod, Apple) \sim (X, Microsoft)$ における最尤の X を求める手法を提案した．[1] はこの問題を Web コーパスを用いた言語横断での潜在比喩検索に適用した．この研究で取り組んだ $(A, B) \sim (C, D)$ は，例えば，$(Japan, Mt.Fuji) \sim (Germany, X)$ における最尤の X を求める手法を提案した．分散表現空間間の変換行列を用いて時代横断の $(A, B) \sim (C, D)$ 問題を解く手法を [9] が提案している．この手法が我々の基礎となる手法である．

3. 問題定義: $(A, B) \sim (C, D)$ 問題としての擬人化生成

一般に，$(A, B) \sim (C, D)$ 問題の入力はエンティティ a，b，および c からなるタプル (a, b, c) である．このタプル (a, b, c) を我々はクエリ q と呼ぶ．この解は出力候補 D の元 d を $(a, b) \sim (c, d)$ の成立尤度に応じて並べた順序集合 $\mathrm{Rank}(q)$ を出力することである．本紙では，a，b，c および d を集合の元とし，以下で定義する．集合 $A = \{s_i\}$，集合 $C = \{\text{“人”}\}$ とする．s_i は文字列であり，主語と呼ぶ．v_j を文字列とし，述語と呼ぶ．$V = \{v_j\}_{j \leq |V|}$（$|V|$ は V の要素数）を v_j の全順序集合とし，述語集合と呼ぶ．$B = D = V$ とする．$\mathbf{Q} = A \times B \times C$ とする．$\mathbf{Q}$ をクエリ集合と呼ぶ．$q = (a, b, c) \in \mathbf{Q}$ とする．$Emb : \mathbf{Q} \to \mathbb{R}^n$ を写像とする．$d_k \in D$ とする．$\mathrm{Sim} : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R} (n \in \mathbb{N})$ は実数値関数であり，(a, b) と (c, d_k) 間の類似度 $\mathrm{rank}(q, d_k)$ を関係 $(a, b) \sim (c, d)$ の成立尤度として返す．$\mathrm{rank}(q, d_k)$ は以下で定義される．

$$\mathrm{rank}(q, d_k) := \mathrm{Sim}(Emb(a, b), Emb(c, d_k)). \quad (1)$$

$\mathrm{rank}(q, d_k)$ により $\{a\} \times D$ を降順にならべかえ全順序集合 $\mathrm{Rank}(q)$ を出力する．特に，$\mathrm{Rank}(q)$ の最初のタプル (a, d_1) を最尤出力と呼ぶ．例えば，入力 q =(飛行船, 飛ぶ, 人) が与えられた時，「飛行船が泳ぐ」を意味する（飛行船，泳ぐ）が最尤出力 $(a, d_1) \in \mathrm{Rank}(q)$ として適切である．

4. 提案手法

提案手法は以下の手順により構成される．

手順 ① 語の分散表現への変換
手順 ② 述語行列の圧縮
手順 ③ 擬人化のベクトル表現
手順 ④ SVD による主成分抽出
手順 ⑤ 分散表現空間間の変換行列
手順 ⑥ 分散表現空間間の類似語のランキングづけと出力

手順の概要を図 1 に示す（図中番号は手順番号に対応）．

手順 ①：語の分散表現への変換 単語の語意表現として，日本語エンティティベクトル（JWiki と後述）を使用した．これは日本語 Wikipedia[*3]に掲載されている語を 200 次元実ベクトル（分散表現）に変換する辞書である．JWiki から以下のように擬人化のベクトル表現を作成した．クエリとして主語とそれに付随する字義的な述語の組 (s_i, v_i) $(i \in \mathbb{N})$（例（“飛行船”，“飛ぶ”））が与えられるとする．JWiki から (s_i, v_i) の分散表現ベクトル $w_v(s_i), w_v(v_i)$ を取得する．さらに，変換行列を求めるための主語 “人” の分散表現ベクトル $w_v(\text{“人”})$ を取得する（$A = \{s_i\}$，$C = \{\text{“人”}\}$）．JWiki の全ての動詞を形態素解析し，動詞の基本形の集合 V と名詞の集合 $n_{l l \leq |N|}$ を抽出する．$w_v(v_i)$ を第 i 行とする行列を V_{base} とし，述語行列と呼ぶ（$B = D = V_{base}$）．$w_v(n_l)$ を第 l 行とする行列を N とし，主語行列と呼ぶ．

[*2] https://wordnet.princeton.edu/

[*3] https://ja.wikipedia.org/wiki/

手順 ②：主語行列と述語行列の圧縮 主語と述語が等価に作用していることは自明ではない．[3] はその Skip-gram の入力の単語出現頻度行列と，この行列を SVD によって分解した後の $U\Sigma$ とがコサインによる類似度に関して同じであることを実験的に示している．これに習い，この主語と述語の作用を調べるために相対的に主語と述語の分散表現の次元の大きさを変えた．具体的には V_{base} を特異値分解（SVD）による主成分分析を行った．SVD は以下のような矩形行列の分解手法である．

$$V_{base} = U\Sigma V^* \tag{2}$$

U, Σ, V^* はそれぞれ V_{base} の出力空間の正規直交基底，V_{base} の特異値を対角成分とする $|V_{base}| \times 200$ の対角行列（$|V_{base}|$ は V_{base} の行数），入力空間の正規直交基底である．$U\Sigma$ の第 1 から第 60 列までとった行列 $V_{\mathrm{SVD}}^{U\Sigma_{60}}$ とする．60 としたのは，1 から 200 までの値のうちで最も MRR が高かったためである．$U\Sigma$ の代わりに U の用いる行列を $V_{\mathrm{SVD}}^{U_{60}}$ とする．$V_{\mathrm{SVD}}^{U\Sigma_{60}}$ または $V_{\mathrm{SVD}}^{U_{60}}$ を区別しない場合は単に V_{SVD} と記す．いずれも $|V_{base}| \times 60$ の行列となる．V_{SVD} の第 i 行を $w_v(v_i)$ の代わりに用いる手法も用いた．同様の圧縮を名詞 N に対して行った行列をそれぞれ $N_{\mathrm{SVD}}^{U\Sigma_{60}}$，$N_{\mathrm{SVD}}^{U_{60}}$ とする．N_{SVD} の第 i 行を $w_v(s_i)$ の代わりに用いる手法も用いた．

手順 ③：擬人化のベクトル表現

すなわち，ベクトル $w_v(s_i) \otimes w_v(v_j)$ の次元は $w_v(s_i)$ の次元 200 と $w_v(v_j)$ の次元 200 の積 40000 である（V_{base} の代わりに V_{SVD} を用いた場合は 12000 次元）．$w_v(s_i) \otimes w_v(v_j)$ を第 j 行とする行列を $w_v(s_i) \otimes V_{\mathrm{base}}$（$V_{\mathrm{SVD}}$ を用いた場合は $w_v(s_i) \otimes V_{\mathrm{SVD}}$）とする．クロネッカー積以外の演算 として，$w_v(s_i)$ と $w_v(v_j)$ の和，各成分の積を成分とするベクトル，並列がありそれぞれ以下のように定義される．これらの演算をクロネッカー積の代わりに用いた手法も用いた．

$$w_v(s_i) + w_v(v_j) = (s_1 + v_1, \ldots, s_{200} + v_{200}), \tag{3}$$

$$w_v(s_i) \times w_v(v_j) = (s_1 v_1, s_2 v_2, \ldots, s_{200} v_{200}), \tag{4}$$

$$(w_v(s_i), w_v(v_j)) = (s_1, \ldots, s_{200}, v_1, \ldots, v_{200}), \tag{5}$$

$$w_v(s_i) \otimes w_v(v_j) = (s_1 v_1, \ldots, s_1 v_{200}, s_2 v_1, \ldots, s_{200} v_{200}), \tag{6}$$

主語，述語の語意をクロネッカー積 $w_v(s_i) \otimes w_v(v_j)$ は以下で定義され，各行を並列に並べたベクトルである．

$$w_v(s_i) \otimes w_v(v_j) = \begin{pmatrix} s_1 v_1 & s_1 v_2 & \cdots & s_1 v_{200} \\ s_2 v_1 & s_2 v_2 & \cdots & s_2 v_{200} \\ \vdots & \vdots & \ddots & \vdots \\ s_{200} v_1 & s_{200} v_2 & \cdots & s_{200} v_{200} \end{pmatrix},$$

$$\tag{7}$$

s_k, v_l はそれぞれ $w_v(s_i)$ の第 k 成分と，$w_v(v_j)$ の第 l 成分である．また，$w_v(s_i) \otimes w_v(v_j)$ と，$w_v(s_i) \otimes w_v(v_j)$ の各行を並列したベクトルを同一視する．

手順 ④：SVD による主成分抽出 JWiki の動詞の基本形の語数 $|V_{SVD}|$ は 6,166 である．$w_v(s_i) \otimes w_v(v_j)$ の次元は 12000 次元であるため $|V_{SVD}|$ を超え，V_{SVD} では後述の変換行列が一意に定まらない．解決策として，手順 ③と同様に主成分分析を行った．

$$w_v(s_i) \otimes V_{\mathrm{SVD}} = U'\Sigma'V'^* \tag{8}$$

$U'\Sigma'_{200}$ の第 1 から第 200 列までとり $w_v(s_i) \otimes V_{\mathrm{SVD}}$ を $|V_{\mathrm{SVD}}| \times 200$ に圧縮した．$U'\Sigma'_{200}$ の代わりに U'_{200} を使用する手法も用いた．$w_v(s_i) \otimes V_{\mathrm{SVD}}$，$w_v(\text{“人”}) \otimes V_{\mathrm{SVD}}$ を圧縮した行列をそれぞれ V_1，V_2 とする．

手順 ⑤：変換行列 V_1 の第 1 行から第 k 行までを V_1^k と表記する．V_2 の第 1 行から第 k 行までを V_2^k と表記する．後述の実験では，k を V_{SVD} の 15% に当たる 924 とした．V_1^k と V_2^k の間の変換行列 M を $w_v(s_i) \otimes V_{\mathrm{SVD}}$ と $w_v(\text{“人”}) \otimes V_{\mathrm{SVD}}$ の間の変換行列とみなす．M を最小二乗法で式（9）を最小化し算出する．

$$M := \arg\min \sum_{i=1}^{k} ||Mu_{1i} - u_{2i}||_2^2 + \gamma ||M||^2. \tag{9}$$

ここで，u_{1i} は V_1^k の第 i 行であり，u_{2i} は V_2^k の第 i 行である．第 2 項は過学習防止のための正規化項である（$\gamma = 0.02$）．

手順 ⑥：分散表現空間間の類似語のランキングづけ 以下に与えられる類似度により，$q = (s_i, v_i, h)$ (h =“人”) を提案手法により検索し，$\{w_v(s_i)\} \times V_{\mathrm{SVD}}$ のランキング $\mathrm{Rank}(q)$ が出力される．$\{w_v(s_i)\} \times V_{\mathrm{SVD}}$ は，「s_i における v_i」が「“人” における d_k」となるような述語を含む最尤出力 (s_i, d_1) の候補の集合である．$\mathrm{Rank}(q)$ 中の d_k の位置 k は以下で定義される類似度により決定される．

$$\begin{aligned} &\mathrm{Sim}\ (Emb_{A\times B}(s_i, v_i), Emb_{C\times D}(h, d_k)) \\ &:= \ \cos(Mw_v(s_i) \otimes w_v(v_i), w_v(h) \otimes w_v(d_k)). \end{aligned} \tag{10}$$

ただし，手法の特性として d_1 が入力の v_i となるため $\mathrm{Rank}(q)$ から省く．

5. 実験

学習データと正解データ JWiki から MeCab[*4]で形態素解析を行い，245,267 語の名詞と 6,166 語の動詞の基本形を抽出した．Web から名詞と動詞の対を 82 用例集め正解データ E とした（例 （“飛行船”, “泳ぐ”））．用例の文脈から読み取れる述語の解釈を各対に付加した（例 （“飛

[*4] https://taku910.github.io/mecab/

表 1　結果

手法	cosine	cosine$U\Sigma_{60}$	l(-)	l($U\Sigma_{60}^2$)	m(-)	m($U\Sigma_{60}^2$)	h(-)	h($U\Sigma_{60}^2$)
MRR	0.114	0.125	0.0434	0.142	0.0976	0.220	0.0689	0.102
手法	k(-, U')	k(-, $U'\Sigma'$)	k(U_{60}, U')	k($U\Sigma_{60}$, U')	k(U_{60}, $U'\Sigma'$)	k($U\Sigma_{60}$, $U'\Sigma'$)	k($U\Sigma_{60}^2$, U')	k($U\Sigma_{60}^2$, $U'\Sigma'$)
MRR	0.0701	0.114	0.0871	0.241	0.125	0.125	**0.253**	0.122

表 2　用例 10 例

主語	述語	解釈	主語	述語	解釈
花火	泳ぐ	飛ぶ	ピアノ	唄う	鳴る
ヒトデ	歩く	泳ぐ	桜	舞う	散る
国	生まれる	始まる	時代	来る	始まる
街	起きる	動く	雨	叩く	鳴る
アイディア	生まれる	生じる	エンジン	叫ぶ	鳴る

表 3　事例：用例（“飛行船”，“泳ぐ”，“飛ぶ”）の 6 位までの出力

r_i	k(-, $U'\Sigma'$)	m($U\Sigma_{60}^2$)	k($U\Sigma_{60}^2$, U')
1	飛行船 飛べる	**飛行船 泳ぐ**	**飛行船 泳ぐ**
2	飛行船 飛ばす	飛行船 飛べる	飛行船 飛ばす
3	飛行船 飛び立つ	飛行船 放り込む	飛行船 飛べる
4	**飛行船 泳ぐ**	飛行船 走り回る	飛行船 飛び立つ
5	飛行船 飛び回る	飛行船 飛び込む	飛行船 浮かぶ
6	飛行船 駆ける	飛行船 投げ捨てる	飛行船 吹く

行船”，“泳ぐ”，“飛ぶ”））．主語と解釈の対をクエリとした（例　（“飛行船”，“飛ぶ”））．表 2 に用例 10 例を示す．

評価手法と結果　以下の MRR により性能を評価した．

$$\mathrm{MRR} = \frac{1}{|E|}\sum_{i=1}^{|E|}\frac{1}{r_i}. \tag{11}$$

$|E|$ は E の要素数を表す．今回は $|E| = 82$ である．r_i は $d_i \in D$ の順位適合性を表す（1000 位以下は $r_i = 1000$）．

表 1 に実験結果を示す．k(a, b) は式 (7) を用いる手法である．a は手順 ②で V_{base} の圧縮の有無である．圧縮しない場合は “-” と記す．圧縮し $V_{\mathrm{SVD}}^{U\Sigma_{60}}$ を用いた場合は $U\Sigma_{60}$，$V_{\mathrm{SVD}}^{U_{60}}$ の場合は U_{60} と記す．$U\Sigma_{60}^2$ は $N_{\mathrm{SVD}}^{U\Sigma_{60}}$ と $V_{\mathrm{SVD}}^{U\Sigma_{60}}$ を用いた場合を指す．b は手順 ④での圧縮の有無である．記号は手順 ④と同様である．$l(a)$ は式 (3)，$m(a)$ は式 (4)，$h(a)$ は式 (5) による手法である．cosine，cosine$U\Sigma_{60}$ はそれぞれ V_{base}，$V_{\mathrm{SVD}}^{U\Sigma_{60}}$ の第 i 行と他の行のコサイン類似度によるランキングの MRR である．k($U\Sigma_{60}^2$, U') が最も優位で MRR= 0.253 あった．MRR の逆数が概ねの出力順位である．したがって，k($U\Sigma_{60}^2$, U') は最尤出力は概ね 4 位である．m($U\Sigma_{60}$) よりも k($U\Sigma_{60}^2$, U') が優位であることはクロネッカー積が有効であることを示す．結果として k($U\Sigma_{60}^2$, $U'\Sigma'$) よりも k($U\Sigma_{60}^2$, U') が優位であった．

事例として用例（“飛行船”，“泳ぐ”，“飛ぶ”）における 6 位までの出力結果を表 3 に示す．最尤出力 “泳ぐ” は k($U\Sigma_{60}^2$, U') の 1 位である．k(-, $U'\Sigma'$) は 4 位である．m($U\Sigma_{60}$) は k($U\Sigma_{60}^2$, U') と同様に 1 位である．しかし，それ以外の出力が “投げ捨てる” など問題の意図にそわない出力が見られる．他の用例に対しても同様の傾向である．

6. おわりに

人間以外のものの動きを人間の動きのように捉える比喩表現として擬人化は日常会話の中でよく見られる表現である．我々は，擬人化を (A, B)∼(C, D) 問題の枠組みで捉え，擬人化の生成を行った．提案手法は, 主語と述語の語意をその分散表現のクロネッカー積により表現した．擬人化の生成手法として語の分散表現空間間の変換行列を求める [9] の手法を基礎としている．提案手法の性能を MRR により評価した．提案手法は評価実験において擬人化の生成に関してクロネッカー積の性質と有用性を示した．

謝辞

本研究の一部は JSPS 科学研究費助成事業 JP16H02906，JP17H00762，JP16H01756，JP18H03243 による助成を受けたものです．ここに記して謝意を表します．

参考文献

[1] Duc, N. T., Bollegala, D. and Ishizuka, M.: Cross-Language Latent Relational Search between Japanese and English Languages Using a Web Corpus, *Proc. of the 11th TALLIP, ACM, Vol.**11**, Issue 3*, pp. 11:1–11:33 (2012).

[2] Kato, M. P., Ohshima, H., Oyama, S. and Tanaka, K.: Query by Analogical Example: Relational Search Using Web Search Engine Indices, *Proc. of the 18th ACM*, pp. 27–36 (2009).

[3] Levy, O. and Goldberg, Y.: Neural Word Embedding as Implicit Matrix Factorization, *Proc of the 27th NIPS*, pp. 2177–2185 (2014).

[4] Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S. and Dean, J.: Distributed Representations of Words and Phrases and their Compositionality, *Proc of the 26th NIPS*, pp. 3111–3119 (2013).

[5] Mikolov, T., Yih, W.-t. and Zweig, G.: Linguistic Regularities in Continuous Space Word Representations, *Proc. of the 11th NAACL*, pp. 746–751 (2013).

[6] Rumelhart, D. E., Hinton, G. E. and Williams, R. J.: Learning Representations by Back-propagating Errors, *Neurocomputing: Foundations of Research*, pp. 696–699 (1988).

[7] Shutova, E., Kiela, D. and Maillard, J.: Black Holes and White Rabbits: Metaphor Identification with Visual Features, *Proc. of the 14th NAACL*, pp. 160–170 (2016).

[8] Turney, P. D. and Littman, M. L.: Corpus-based Learning of Analogies and Semantic Relations, *Proc. of ML, Vol. 60, Issue 1–3*, pp. 251–278 (2005).

[9] Zhang, Y., Jatowt, A., Bhowmick, S. and Tanaka, K.: Omnia Mutantur, Nihil Interit: Connecting Past with Present by Finding Corresponding Terms across Time, *Proc. of the 53th ACL, Vol. 1:*, pp. 645–655 (2015).