

Learning Composition Models for Phrase Embeddings

Mo Yu, Mark Dredze

TACL2015

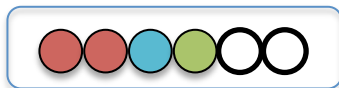
読む人：東北大学，高瀬翔

word embedding

- 単語の特徴（意味）を低次元のベクトルで表現



peach



penguin



drug



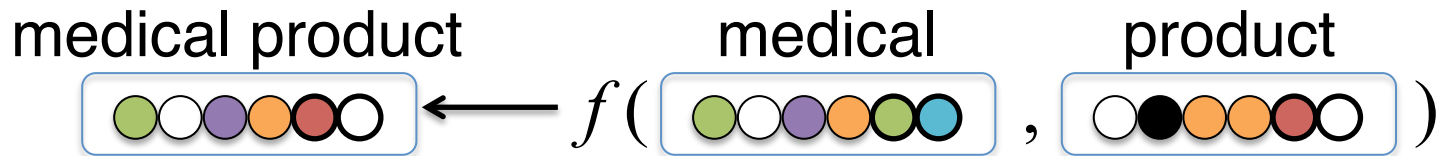
medicine



- どのように得るか？
 - 次元圧縮（e.g., 単語共起行列をSVD）
 - ニューラル言語モデル（e.g., word2vec）

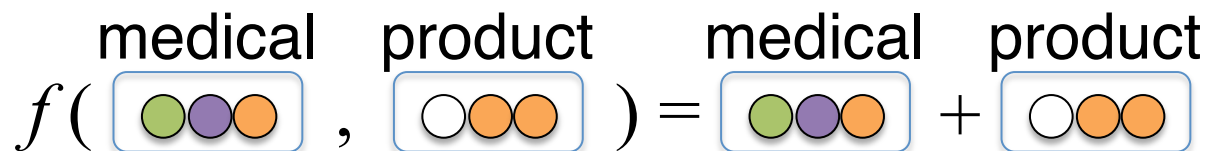
本論文の目的

- フレーズの意味を表す低次元のベクトルを単語のベクトルから構築する



既存研究と問題点

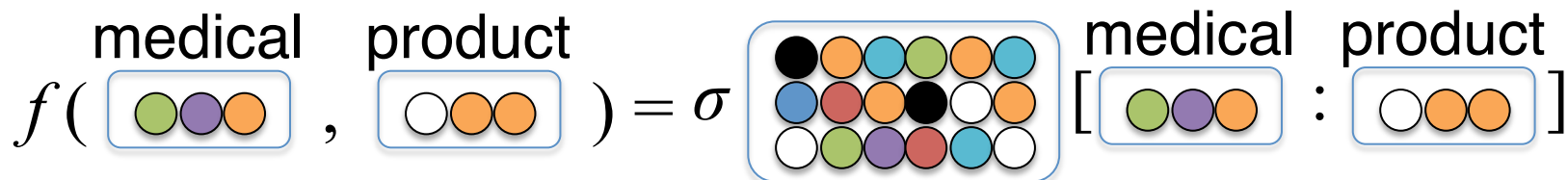
- あらかじめ演算を定義 (e.g., sum)

$$f(\text{medical}, \text{product}) = \text{medical} + \text{product}$$


☹️ 単語の特徴や文脈に適した演算を行えない

- DT (e.g., a, the, this) の意味は無視して良いはず

- 行列やテンソルを利用 (e.g., RecursiveNN)

$$f(\text{medical}, \text{product}) = \sigma \left[\begin{array}{cccccc} \bullet & \bullet & \bullet & \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet & \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet & \bullet & \bullet & \bullet \end{array} \right] [\text{medical} : \text{product}]$$


☹️ 計算量が大きい

- 次元数を増やしづらい
- 小規模なデータセットでの教師あり学習のみ

本研究の概要

- 単語の特徴や文脈に応じた演算でフレーズのベクトルを計算する手法を提案
- 提案手法の計算量は小さい
 - 高次元 (e.g., 200次元) なベクトルも扱える
 - 大規模なデータで学習可能
- 教師なし, 教師あり, 組み合わせで学習, 評価

提案モデル

- フレーズのベクトル：単語ベクトルの重み付き和

$$e_p = \sum_i^N \lambda_i \odot e_{w_i}$$

フレーズのベクトル λ_i 単語 w_i への重みベクトル e_{w_i} 単語ベクトル

- 重みは単語の素性 (e.g., 品詞, 単語の位置) から計算

$$\lambda_{ij} = \sum_k \alpha_{jk} f_k(w_i, p) + b_{ij}$$

λ_{ij} 単語 w_i への重みベクトルの j 次元 $f_k(w_i, p)$ 素性ベクトル

素性

品詞による意味の強さを捉えたい

(DT (e.g., a, the, this) の意味は無視 (重み 0) する)

似た意味の単語は同じ重みで計算して欲しい (big, large, hugeは同じ重みになって欲しい)

Simple Features	
POS tags	$t(w_{i-1}), t(w_i), t(w_{i+1})$
Word clusters	$c(w_{i-1}), c(w_i), c(w_{i+1})$
Head word	w_{i-1}, w_i, w_{i+1} if w_i is function word $I[i = h]$
Distance from head	$\text{Dis}(h - i)$
Head tag/cluster	$t(w_h), c(w_h)$ if $i \neq h$

- 組み合わせ素性は割愛

目的関数

- 教師なし学習
 - skip-gramの目的関数をフレーズに拡張
- 教師あり学習
 - softmaxを用いたmulti label分類
- 2つの組み合わせ

skip-gram[Mikolov+ 13]

- 単語 w_i から周辺語 w_{i+j} の予測確率を最大化
– 周辺単語のベクトルに似るように学習



周辺単語を予測

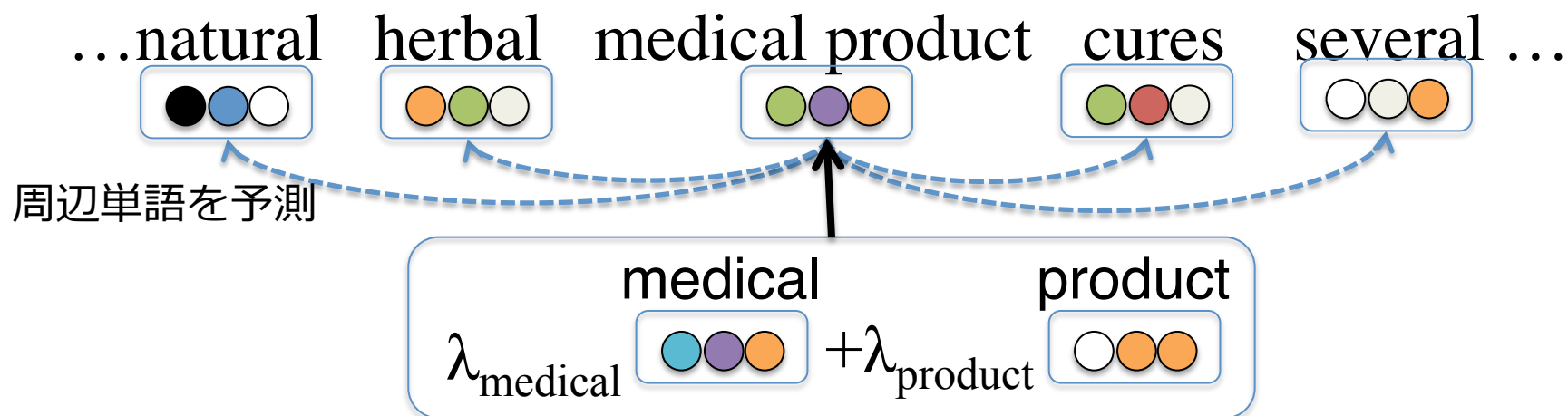
$$J = \sum_{i=1}^S \sum_{c \leq j \leq c, j \neq 0} \log P(w_{i+j} | w_i)$$

文脈の範囲

$$\text{ただし } P(w_{i+j} | w_i) = \frac{\exp(e_{w_{i+j}}^T e_{w_i})}{\sum_{w' \in V} \exp(e_{w'}^T e_{w_i})}$$

skip-gramのフレーズへの拡張

- 単語ベクトルの代わりにフレーズのベクトルで周辺語を予測
 - フレーズのベクトル：単語ベクトルから構築



誤差逆伝播でパラメータ
(単語のベクトル, a , b)
を学習

$$\lambda_{ij} = \sum_k \alpha_{jk} f_k(w_i, p) + b_{ij}$$

softmaxを用いたmulti label分類

- フレーズ p_s がフレーズ p_i と似ているか、分類するタスク
 - p_s が p_i と似ている : $y_i = 1$
 - p_s が p_i と似ていない : $y_i = 0$
 - として

$$\text{目的関数} : \max_{\alpha, \mathbf{b}, \mathbf{e}_w} \sum_{p_s} \sum_{i=1}^N y_i \log P(y_i = 1 | p_s, p_i)$$

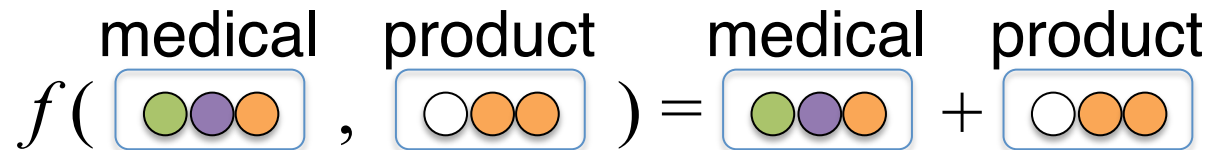
$$= \max_{\alpha, \mathbf{b}, \mathbf{e}_w} \sum_{p_s} \sum_{i=1}^N y_i \log \frac{\exp(e_{p_s}^T e_{p_i})}{\sum_j \exp(e_{p_s}^T e_{p_j})}$$

実験設定

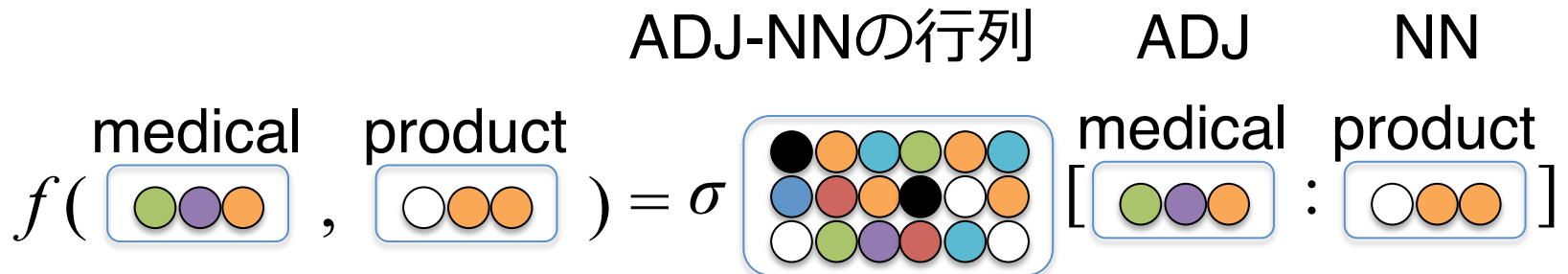
- コーパス：NYT 1994-97 (LDC2011T07)
 - 515,301,382 tokens
 - 語彙：518,235語 (頻度5以上の単語)
- フレーズ：NPとなるbi-gram
 - new trial, dead body, an extension, ...
- ベクトルの次元数：200
- 提案手法の初期値：skip-gramで学習したベクトル
 - skip-gramモデルを比較する際はコーパスを2周する
- skip-gramと提案手法での学習設定 (窓幅や負例のサンプリング数) は合わせる

比較手法

- skip-gramで得たベクトルの和 (SUM)

$$f(\text{medical}, \text{product}) = \text{medical} + \text{product}$$


- RecursiveNN (行列は品詞組み合わせ毎)
 - ADJ-NNの行列, NN-NNの行列, ...

$$f(\text{medical}, \text{product}) = \sigma \begin{matrix} \text{ADJ-NNの行列} & \text{ADJ} & \text{NN} \\ \begin{matrix} \text{medical} & \text{product} \\ \begin{bmatrix} \text{green} & \text{purple} & \text{orange} \\ \text{white} & \text{orange} & \text{orange} \end{bmatrix} \end{matrix} \end{matrix}$$


タスク

- 複数の教師ありタスクで実験
- PPDB : inputに対し, outputとの類似度が候補中で高いか
- SemEval2013 : 2つの表現が類似か否か
- Turney2012 : inputに対し, 正しいoutputを選択できるか
 - 候補中で正解の類似度が最も高くなるか

Data Set	Input	Output
(1) PPDB	medicinal products	drugs
(2) SemEval2013	<small spot, flect> <male kangaroo, point>	True False
(3) Turney2012	monosyllabic word	monosyllable , hyalinization, fund, gittern, killer
(4) PPDB (ngram)	contribution of the european union	eu contribution

PPDB

目的関数

(LM, - : skip-gram

TASK-SPEC : 教師あり学習)

単語ベクトルを

更新するか

正解のランク

の逆数の和

単語毎の重み
ベクトルを学習
(提案手法で
素性ベクトルを
利用しない手法)

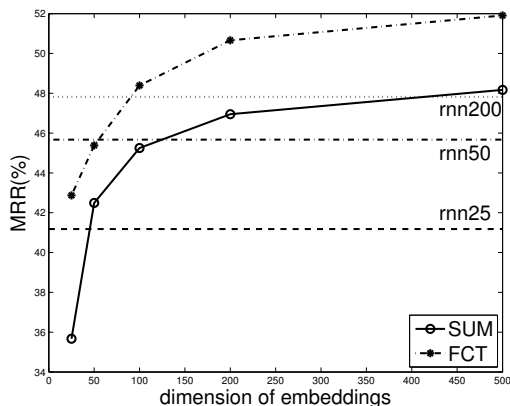
Model	Objective	Fine-tuning Word Emb	MRR @ 10k
SUM	-	-	41.19
SUM	TASK-SPEC	Y	45.01
WSum	TASK-SPEC	Y	45.43
RNN 50	TASK-SPEC	N	37.81
RNN 50	TASK-SPEC	Y	39.25
RNN 200	TASK-SPEC	N	41.13
RNN 200	TASK-SPEC	Y	40.50
FCT	TASK-SPEC	N	41.96
FCT	TASK-SPEC	Y	46.99
FCT	LM	Y	42.63
FCT-P	TASK-SPEC+LM	Y	49.44
FCT-J	TASK-SPEC+LM	joint	51.65

提案手法

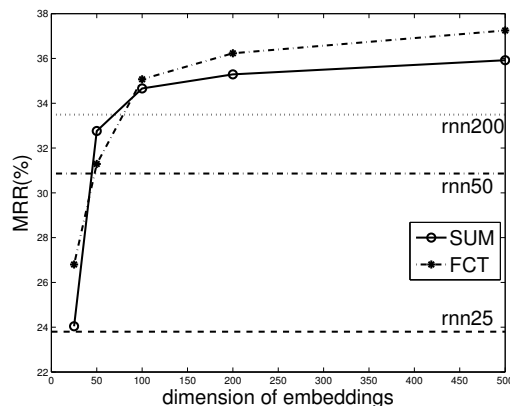
Turney2012, SemEval2013

Model	Objective	Fine-tuning Word Emb	SemEval2013	Turney2012		
			Test	Acc (5)	Acc (10)	MRR @ 10k
SUM	-	-	65.46	39.58	19.79	12.00
SUM	TASK-SPEC	Y	67.93	48.15	24.07	14.32
Weighted Sum	TASK-SPEC	Y	69.51	52.55	26.16	14.74
RNN (d=50)	TASK-SPEC	N	67.20	39.64	25.35	1.39
RNN (d=50)	TASK-SPEC	Y	70.36	41.96	27.20	1.46
RNN (d=200)	TASK-SPEC	N	71.50	40.95	27.20	3.89
RNN (d=200)	TASK-SPEC	Y	72.22	42.84	29.98	4.03
Dual Space ¹	-	-	52.47	27.55	16.36	2.22
Dual Space ²	-	-	-	58.3	41.5	-
RAE	auto-encoder	-	51.75	22.99	14.81	0.16
FCT この2つは	TASK-SPEC	N	68.84	41.90	33.80	8.50
FCT 50次元かも	TASK-SPEC	Y	70.36	52.31	38.66	13.19
FCT	LM	-	67.22	42.59	27.55	14.07
FCT-P	TASK-SPEC+LM	Y	70.64	53.09	39.12	14.17
FCT-J	TASK-SPEC+LM	joint	70.65	53.31	39.12	14.25

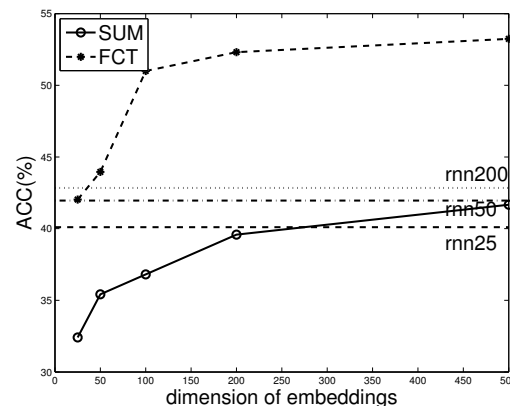
ベクトルの次元と性能



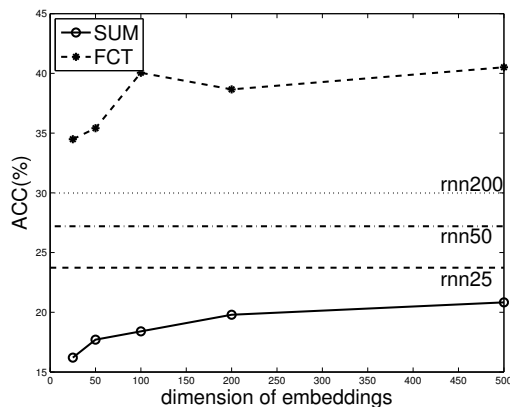
(a) MRR@1k on PPDB dev set



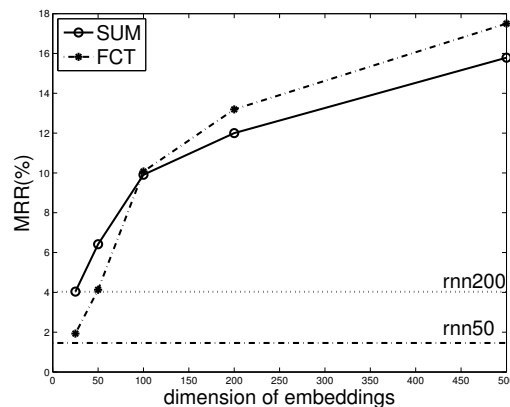
(b) MRR@10k on PPDB dev set



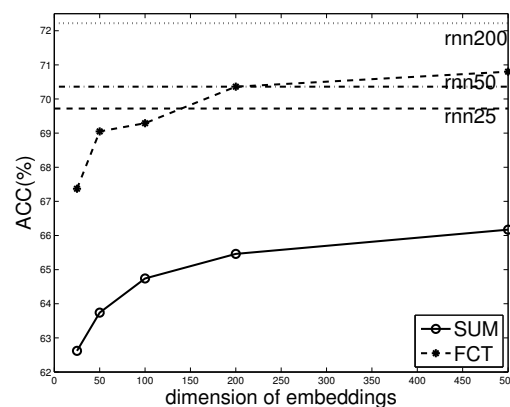
(c) accuracy on the 5-choice task in Turney2012



(d) accuracy on the 10-choice task in Turney2012



(e) MRR@10k on Turney2012



(f) accuracy on the SemEval2013

結論

- 単語の特徴や文脈に応じた演算でフレーズのベクトルを計算する手法を提案
- 単語ベクトルの和, RNNよりも良い性能であると示した
 - 教師なし学習（大量のデータ使用）を組み合わせるとさらに良くなる

おまけ：計算量，計算時間

- 単語ベクトルの次元数200での訓練時に
 - 提案手法：2.33 instance / ms
 - RNN：0.31 instance / ms
- と論文では報告している
- しかし，計算量は
 - 提案手法： $O(\text{発火した素性数} * \text{次元数}^2)$
 - 重みベクトルの計算量： $O(\text{発火した素性数} * \text{次元数})$
 - RNN： $O(\text{次元数}^2)$
- に思うので，ちょっと良くわからない

おまけ：素性と性能の変化

Feature Set	MRR @ 10k
FCT	79.68
-clus	76.82
-POS	77.67
-Compound	79.40
-Head	77.50
-Distance	78.86
WSum	75.37
SUM	74.30

- 単語クラスタが最も効果がある
- 単語ごとに重みベクトルを学習 (WSUM) は低い
 - クラスタにしないと疎だから？
 - 素性は前後の単語も見ているから？

おまけ : skip-gramモデルの perplexity

Model	Perplexity (HS training)			NCE loss (NCE training)		
	Subset Train	Dev	Test	Subset Train	Dev	Test
SUM (2 epochs)	7.620	7.577	7.500	2.312	2.226	2.061
word2vec (2 epochs)	7.103	7.074	7.052	2.274	2.195	2.025
FCT (random init, 2 epochs)	6.753	6.628	6.713	1.879	1.722	1.659
FCT (with pre-training, 1 epochs)	6.641	6.540	6.552	1.816	1.691	1.620

- 提案手法はperplexity, lossが低いのでフレーズのベクトルから周辺語の予測が良く出来ている
- 学習時にフレーズを学習するかで窓幅が変わる可能性があり, 公平な比較が少し疑問
 - herbal medical product curesについて
 - skip-gram : medicalの周辺N単語
 - 提案手法 : medical productの周辺N単語