Improving Distributional Similarity with Lessons Learned from Word Embeddings

Omer Levy, Yoav Goldberg, Ido Dagan
(Bar-Ilan University)
[TACL 2015]

発表: 原　忠義 （東京大学 坂田・森研究室）

前置き

- 時間の都合で内容をかなり割愛しています
  - 多少補足を付けましたので、発表後ご質問 or 必要に応じご確認いただければ…
  - 人工知能学会での岡崎先生の講演資料
    http://www.slideshare.net/naoakiokazaki/20150530-jsai2015
    - 本論文含む近年の分散表現研究を整理
    - 本紹介が解りづらいようでしたらご参考に…

概要:
単語分散表現モデルの性能比較

【対象】: 4モデル

- ◆ ニューラルモデル（SGNS, GloVe）
- ◆ 頻度ベースモデル (PPMI, SVD)

【知見】: ニューラルモデル の優位性は「ハイパーパラメータの最適化」によるもの

- ◆ ニューラルモデルのパラメータを頻度ベースモデルにも導入することで同等の性能に
  - ● global にはアプローチの有利不利はない

比較する4手法（モデル）

- 頻度ベース表現モデル
  - Positive Pointwise Mutual Information (**PPMI**) 行列
  - Singular Value Decomposition (**SVD**)
- ニューラルモデル
  - Skip-grams with negative sampling (**SGNS**)
  - Global Vectors (**GloVe**)
- どれも単語を bag of context-words で表現
  - シンプルかつ高性能*

---

* [Mikolov et al. 2013, Pennington et al., 2014]

対象モデル (1/4): Positive Pointwise Mutual Information (**PPMI**) 行列*

- $M_{i,j}^{PPMI} = PPMI(w_i, c_j) = \max(PMI(w_i, c_j), 0)$
  - $PMI(w_i, c_j) = \log \frac{\hat{P}(w_i, c_j)}{\hat{P}(w_i)\hat{P}(c_j)}$
  - 対象単語 w と文脈 c の相互情報量
  - 0との max をとることで正の値のみに
  - 【弱点】希な文脈 c で頻度少の w の影響大

負の発散を防ぐ

↓w　c→	have	new	drink	bottle	ride	speed	read
beer	0	0	2.04	1.97	0	0	0
car	0.09	0.49	0	0	0.13	0.55	0

* [Bullinaria and Levy, 2007]

対象モデル (2/4): Singular Value Decomposition (**SVD**)*

- M^{PPMI}を rank d の密な空間へ低次元化
- $M^{PPMI} = U \cdot \Sigma \cdot V^{\intercal}$
  - U, V: 直交行列
  - Σ: 固有値対角行列 → Σ_d: ランク d に限定
- $M_d^{SVD} = U_d \cdot \Sigma_d \cdot V_d^{\intercal}$
  - $W^{SVD} = U_d \cdot \Sigma_d, \quad C^{SVD} = V_d$
- 計算効率向上, より一般化

---

* [Deerwester et al., 1990]

対象モデル (3/4): Skip-grams with negative sampling (**SGNS**)*1

- ニューラルモデル
  - 単語と文脈を d-次元ベクトル $\vec{w}, \vec{c}$ で表現
- 観測 $\vec{w} \cdot \vec{c}$ を最大／非観測 $\vec{w} \cdot \overrightarrow{c_N}$ を最小に
  - c_N は c の分布から k 個（サンプル）作成
    - unigram分布 $P_D(c) = \#(c)/|D|$ (D:ドメイン)
- word2vec において実装*1
- 【特徴】Shifted PMI*2
  - $\vec{w} \cdot \vec{c} = \mathrm{PMI}(w, c) - \log k$ で目的関数最適化
  - $W \cdot C^{\intercal}$ を $\log k$ でシフトした M^{PMI} で分解

*1 Mikolov et al., 2013 *2 Levy and Goldberg, 2014

対象モデル (4/4): Global Vectors (**GloVe**)*

- ニューラルモデル, 単語と文脈: d-次元
- 単語・文脈ベクトルは以下を充足
  - $\vec{w} \cdot \vec{c} + b_w + b_c = \log(\#(w, c))$
  - $b_w,\ b_c$: バイアス項 ←**自由度追加**
- 目的関数:頻度対数化行列＋バイアス項
  - $M^{\log(\#(w,c))} \approx W \cdot C^{\intercal} + \overrightarrow{b^w} + \overrightarrow{b^c}$
- 単語 w : context ベクトルとその context における単語ベクトルの和 $(\vec{c} + \vec{w})$ で表現

---

* Pennington et al., 2014

モデル性能の従来知見に対する疑問

- ニューラル ＞ 頻度ベース *1 … ？
- GloVe ＞ SGNS *2 … ？
- 数学的な目的や利用情報は似通っている
  - 単語の bag-of-contexts 表現に基づく
  - SGNS は暗に単語-文脈 PMI行列を分解

→ モデルの差以外に、明示・非明示的に調整されているパラメータが影響しているのでは？

各モデルで共通化・調整可能にし、性能比較を

*1 Baroni et al., 2014

*2 Pennington et al., 2014

性能比較の流れ

【準備】: ハイパーパラメータの共通化

- ◆ ニューラルモデル側のパラメータを明示化
  → 従来モデルでも調整できるよう導入

【実施】: 複数タスクセット上での比較検証

- ◆モデルの（最適パラメータでの）性能比較
- ◆各パラメータの有効性検証

【まとめ】:

- ◆実用上おすすめのモデル・パラメータ選択

性能比較の流れ

【準備】: ハイパーパラメータの共通化

◆ ニューラルモデル側のパラメータを明示化

→ 従来モデルでも調整できるよう導入

【実施】: 複数タスクセット上での比較検証

◆モデルの（最適パラメータでの）性能比較

◆各パラメータの有効性検証

【まとめ】:

◆実用上おすすめのモデル・パラメータ選択

性能比較の流れ

【準備】: ハイパーパラメータの共通化

- ◆ ニューラルモデル側のパラメータを明示化
  → 従来モデルでも調整できるよう導入

【実施】: 複数タスクセット上での比較検証

- ◆モデルの（最適パラメータでの）性能比較
- ◆各パラメータの有効性検証

【まとめ】:

- ◆実用上おすすめのモデル・パラメータ選択

実験設定 (1/2): 単語表現の学習

- コーパス: 英語 Wikipedia (2013/8 dump)
  - 非テキスト除外・文区切り・tokenization
  - 7750万文、15億トークン
  - 出現100回未満の単語は無視 → 189,533語
- SVD, SGNS, GloVe: 500次元で学習
- SGNS: word2vec（の改造版）を使用
- GloVe: オリジナル実装を用いて 50 iteration

実験設定 (2/2):タスクセット（**単語類似度**／~~Analogy~~）

割愛

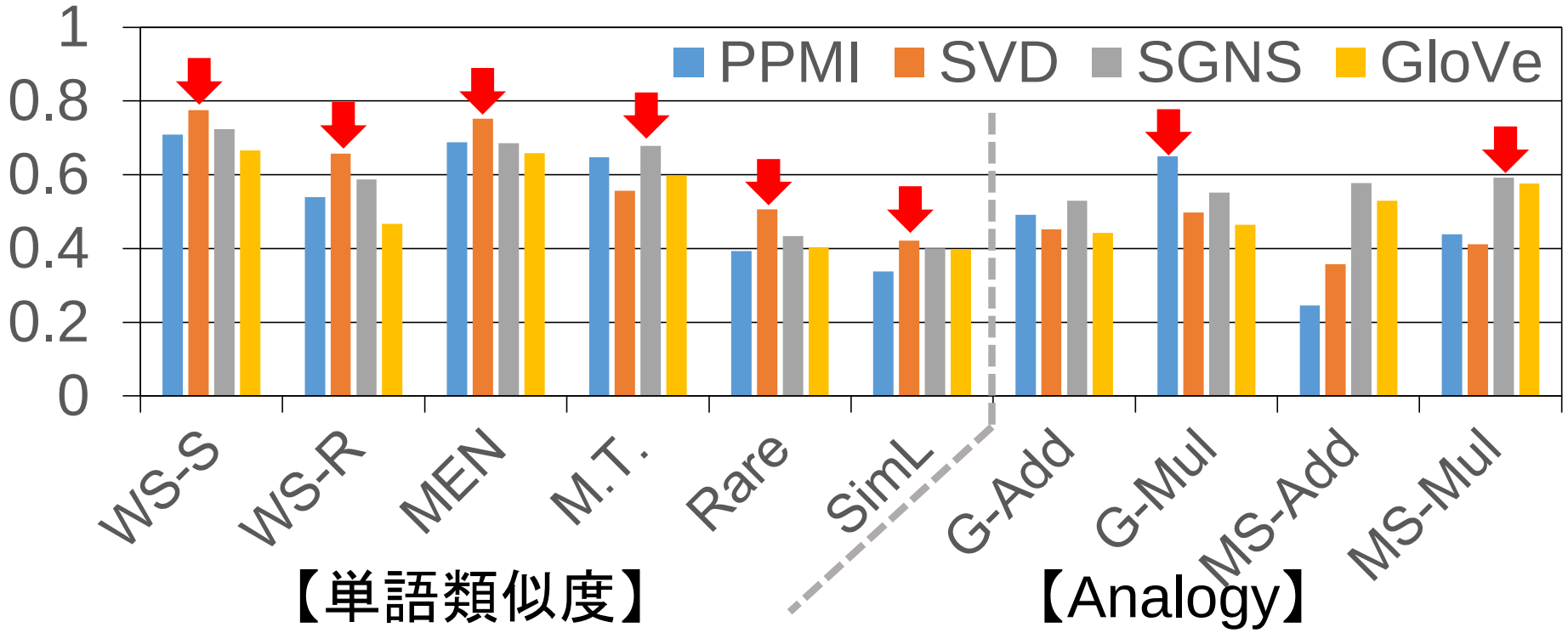
- 6種類（人手による類似度スコア付）
  - WordSim353*[1] Similarity*[2] (“WS-S”)
  - WordSim353 Relatedness*[2] (“WS-R”)
  - MEN*[3] (“MEN”)
  - Mechanical Turk*[4] (“M. T.”)
  - Rare Words*[5] (“Rare”)
  - SimLex-999*[6] (“SimL”)
- 評価：単語ペアをコサイン類似度で順位付
  →人間判定との相関関係（Spearman’s ρ）

---

*[1]Finkelstein et al., 2002 *[2]Zesch et al., 2008他 *[3]Brui et al., 2012,
*[4]Radinsky et al., 2013, *[5]Luong et al., 2013, *[6]Hill et al., 2014

↓ 各データセットでのベスト性能

性能比較 (1/3):
【 頻度ベースモデルに近い設定＊】

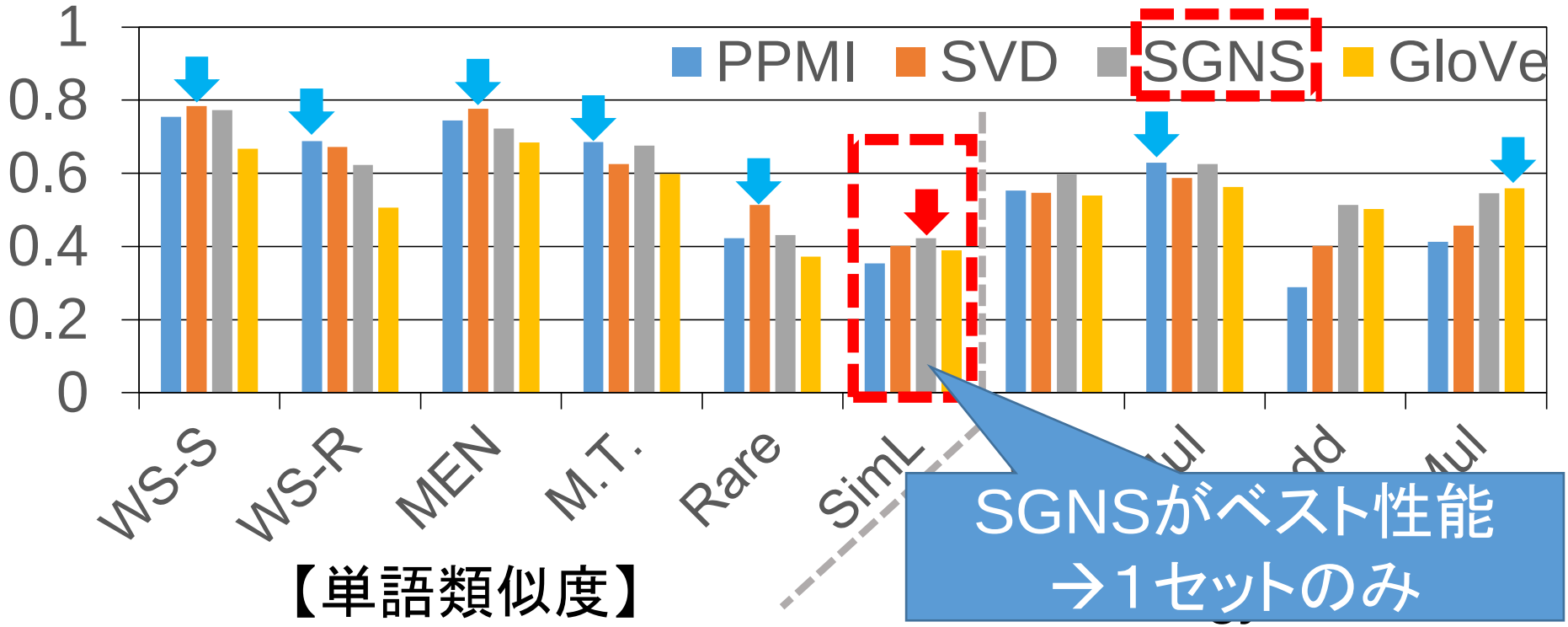


【＊設定値（各詳細は後ほど）】
context window (**win**) = 2
dynamic context (**dyn**) = none
subsampling (**sub**) = none

negative samples (**neg**) = 1
context smoothing (**cds**) = 1
add context vector (**w+c**) = only w
eigenvalue weight (**eig**) = 0.0

SGNSがベストを達成　SGNS以外がベストを達成

性能比較 (2/3): 【 word2vec (SGNS) での設定＊】



【単語類似度】

【＊設定値（各詳細は後ほど）】

context window (**win**) = 2

dynamic context (**dyn**) = **with**

subsampling (**sub**) = **dirty**

negative samples (**neg**) = **5**

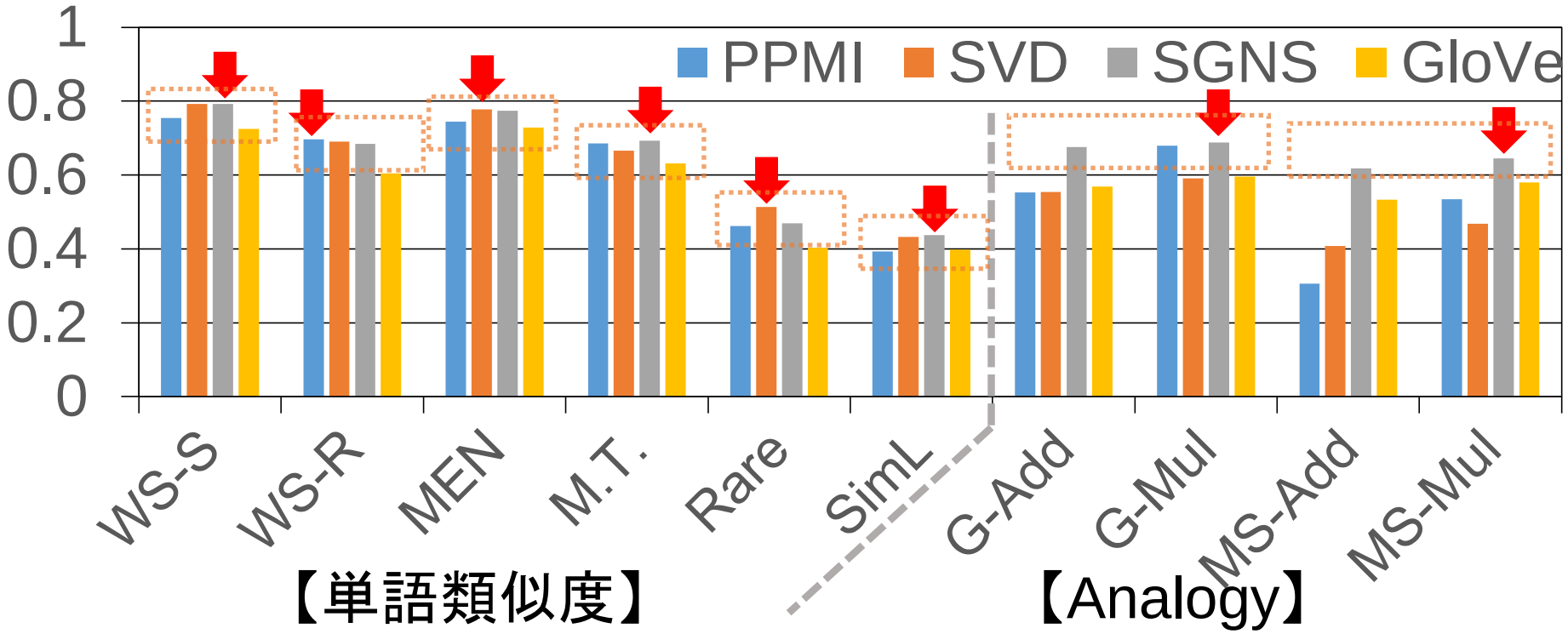
context smoothing (**cds**) = **0.75**

add context vector (**w+c**) = only *w*

eigenvalue weight (**eig**) = 0.0

各データセットでのベスト性能

性能比較 (3/3):
【 win = 2 以外をモデル毎に調整】



頻度ベース設定 → word2vec 設定 → 個別調整 で
性能の大幅改善（最大 +0.157, 平均 +0.06）

モデル選択よりパラメータ調整の方がインパクト大

従来の知見は正しいのか？

- **ニューラルモデル ＞ 頻度ベース** *1 → △
  - × 類似度： SGNS平均 ＜ SVD平均
  - ○ Analogy: [SGNS, GloVe] ＞＞ [PPMI, SVD]
- **GloVe ＞ SGNS** *2 → ×
  - Analogy の1件を除いて全て SGNS ＞ GloVe

*1 Baroni et al., 2014
*2 Pennington et al., 2014
*3 Levy & Goldberg, 2014

性能比較の流れ

【準備】: ハイパーパラメータの共通化

- ◆ ニューラルモデル側のパラメータを明示化
  → 従来モデルでも調整できるよう導入

【実施】: 複数タスクセット上での比較検証

- ◆モデルの（最適パラメータでの）性能比較
- ◆各パラメータの有効性検証

【まとめ】:

- ◆実用上おすすめのモデル・パラメータ選択

手法間で共通化された ハイパーパラメータ空間

	パラメータ名	確かめた設定値	導入可能なモデル
	win	2, 5, 10	全モデル
前処理	**dyn**	none, with	全モデル
前処理	**sub**	none, dirty, clean	全モデル
前処理	**del**	none, with	全モデル
計算	**neg**	1, 5, 15	PPMI, SVD, SGNS
計算	**cds**	1, 0.75	PPMI, SVD, SGNS
後処理	**w+c**	only w, $w + c$	SVD, SGNS, GloVe
後処理	**eig**	0, 0.5, 1	SVD
後処理	**nrm**	none, row, col, both	全モデル

手法間で共通化された ハイパーパラメータ空間

	パラメータ名	確かめた設定値	導入可能なモデル
	win	2, 5, 10	全モデル
前処理	**dyn**	none, with	全モデル
前処理	**sub**	none, dirty, cle	
前処理	del	none, with	全モデル
計算	**neg**	1, 5, 15	PPMI, SVD, SGNS
計算	**cds**	1, 0.75	PPMI, SVD, SGNS
後処理	**w+c**	only w, $w + c$	SVD, SGNS, GloVe
後処理	**eig**	0, 0.5, 1	SVD
後処理	nrm	none, row, col, both	全モデル

事前実験で効果なし
（説明は割愛）

手法間で共通化された ハイパーパラメータ空間

	パラメータ名	確かめた設定値	導入可能なモデル
	win	2, 5, 10	全モデル
前処理	**dyn**	none, with	全モデル
	sub	none, dirty, clean	全モデル
	del	none, with	全モデル
計算	**neg**	1, 5, 15	PPMI, SVD, SGNS
	cds	1, 0.75	PPMI, SVD, SGNS
後処理	**w+c**	only w, $w + c$	SVD, SGNS, GloVe
	eig	0, 0.5, 1	SVD
	nrm	none, row, col, both	全モデル

Context Window Size とその効果

- **Context Window Size**: **win** = [2 / 5 / 10]
  - 単語の前後 **win** 単語ずつを context にする
- 【知見】: PPMI,SVD は狭めが良い？

↓ ベスト性能モデルにおける Window Size（8テスト内訳）

	win = 2	win = 5	win = 10	計テスト数
PPMI	**7**	1	0	8
SVD	**7**	1	0	8
SGNS	2	3	3	8
GloVe	1	3	4	8

手法間で共通化された ハイパーパラメータ空間

	パラメータ名	確かめた設定値	導入可能なモデル
	win	2, 5, 10	全モデル
前処理	**dyn**	none, with	全モデル
前処理	**sub**	none, dirty, clean	全モデル
前処理	**del**	none, with	全モデル
計算		1, 5, 15	PPMI, SVD, SGNS
計算		, 75	PPMI, SVD, SGNS
後処理	**w+c**	only w, $w + c$	SVD, SGNS, GloVe
後処理	**eig**	0, 0.5, 1	SVD
後処理	**nrm**	none, row, col, both	全モデル

入力データに影響

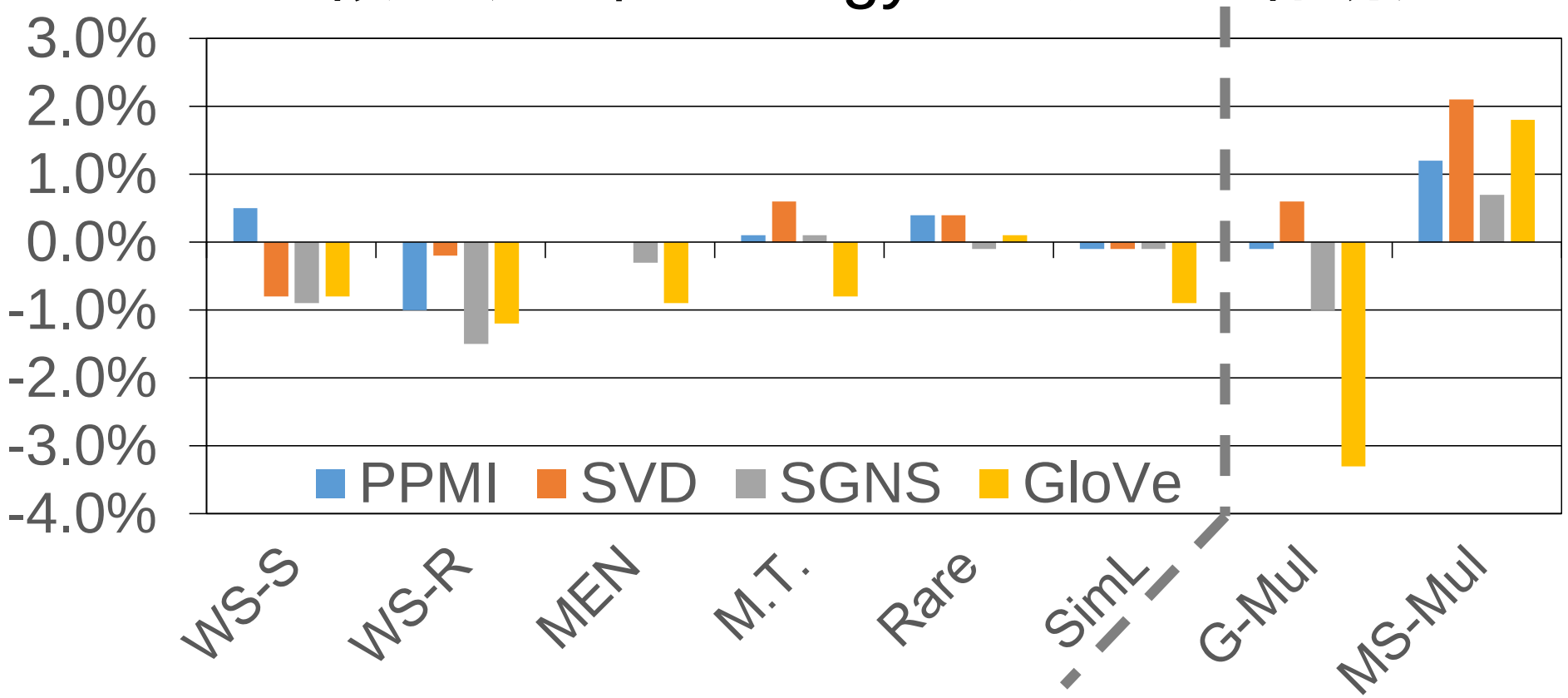
前処理における共通化：SGNS のパラメータ → 全手法へ

- **Dynamic context window**：**dyn** = [none/with]
  - 各トークン毎に context を 1~window size (**win**) でサンプリング→ 距離に基づく重みを動的に決定
- **Subsampling**: **sub** = [none/dirty/clean]
  - 頻度f ≥閾値tの単語を確率$1-\sqrt{t/f}$で除去*
  - context 取得前(dirty)・後(clean)実施で性能差は見られず → $t = 10^{-5}$, dirty（word2vec 準拠）
- **Deleting Rare Words**: **del** = [none/with]
  - 希単語の除去（事前実験で効果なし→カット）

*word2vec では確率 $(f-t)/f-\sqrt{t/f}$ で除去

前処理パラメータの効果 (1/2): Dynamic Context Window

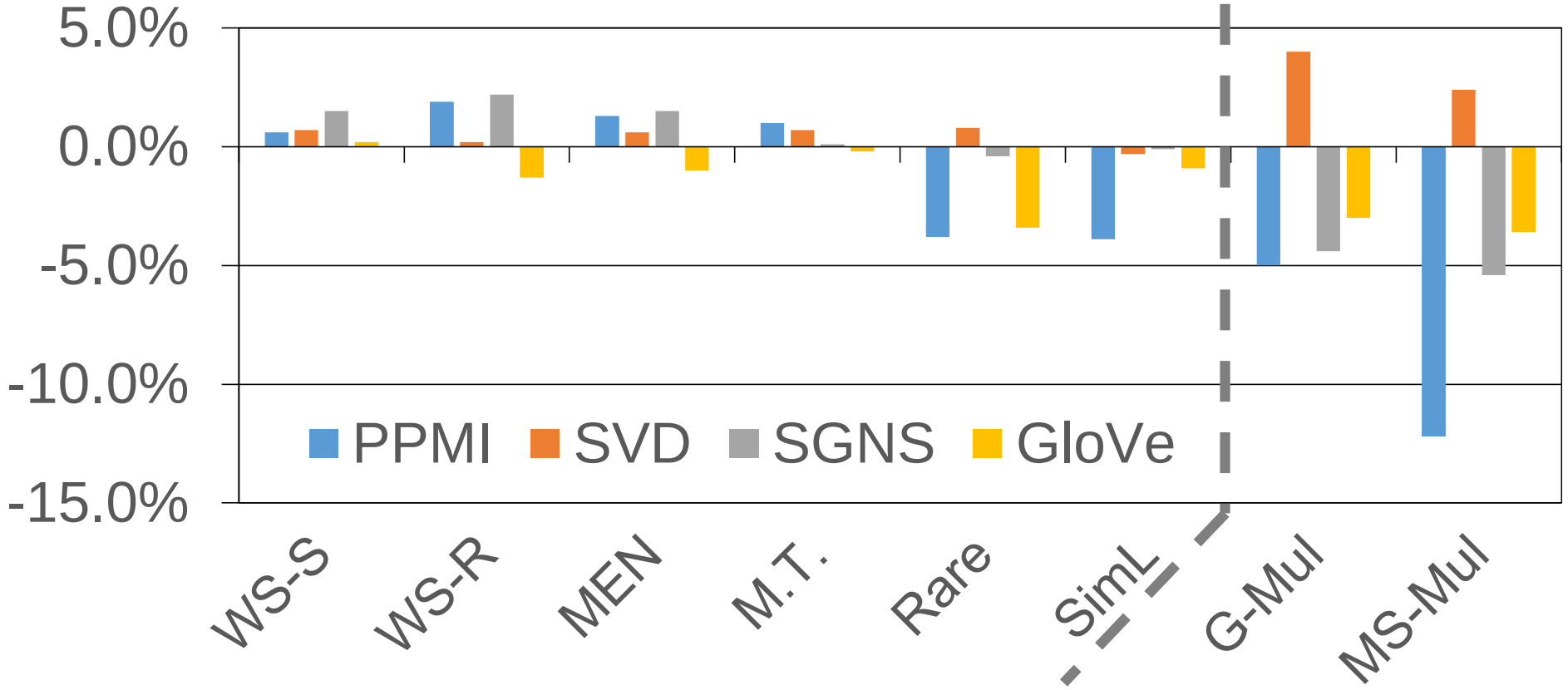
■ 悪化傾向（一部 Analogy タスクには有効）



↑ **dyn** = none → with での変化（他パラメータは最適化）

前処理パラメータの効果 (2/2): Subsampling

■（言及はないが…）SVD には割と有効？



↑ **sub** = none → dirty での変化（他パラメータは最適化）

手法間で共通化された ハイパーパラメータ空間

	パラメータ名	確かめた設定値	導入可能なモデル
前処理	**win**	2, 5, 10	全モデル
			全モデル
			全モデル
	del	none, with	全モデル
計算	**neg**	1, 5, 15	PPMI, SVD, SGNS
	cds	1, 0.75	PPMI, SVD, SGNS
後処理	**w+c**	only w, $w + c$	SVD, SGNS, GloVe
	eig	0, 0.5, 1	SVD
	nrm	none, row, col, both	全モデル

単語-文脈の関連度計算

関連度計算における共通化: SGNS から PPMI (SVD)への導入

- **Shifted PMI**: **neg** = [1 / 5 / 15]
  - 負例サンプリング [PMI (w,c) – log k] を PPMI へ
  - Shifted PPMI (SPPMI):
    - SPPMI (w,c) = max ((PMI(w,c) – log k), 0)
- **Context Distribution Smoothing**: **cds** = [1 / 0.75]
  - 全ての context count を α乗で底上げ*
    - 希単語に対する PMI の偏重を軽減 → PPMIにも
  - $PMI_\alpha(w,c) = \log(\hat{P}(w,c)/\hat{P}(w)\widehat{P_\alpha}(c))$
    - $\widehat{P_\alpha}(c) = \#(c)^\alpha / \sum_c \#(c)^\alpha$

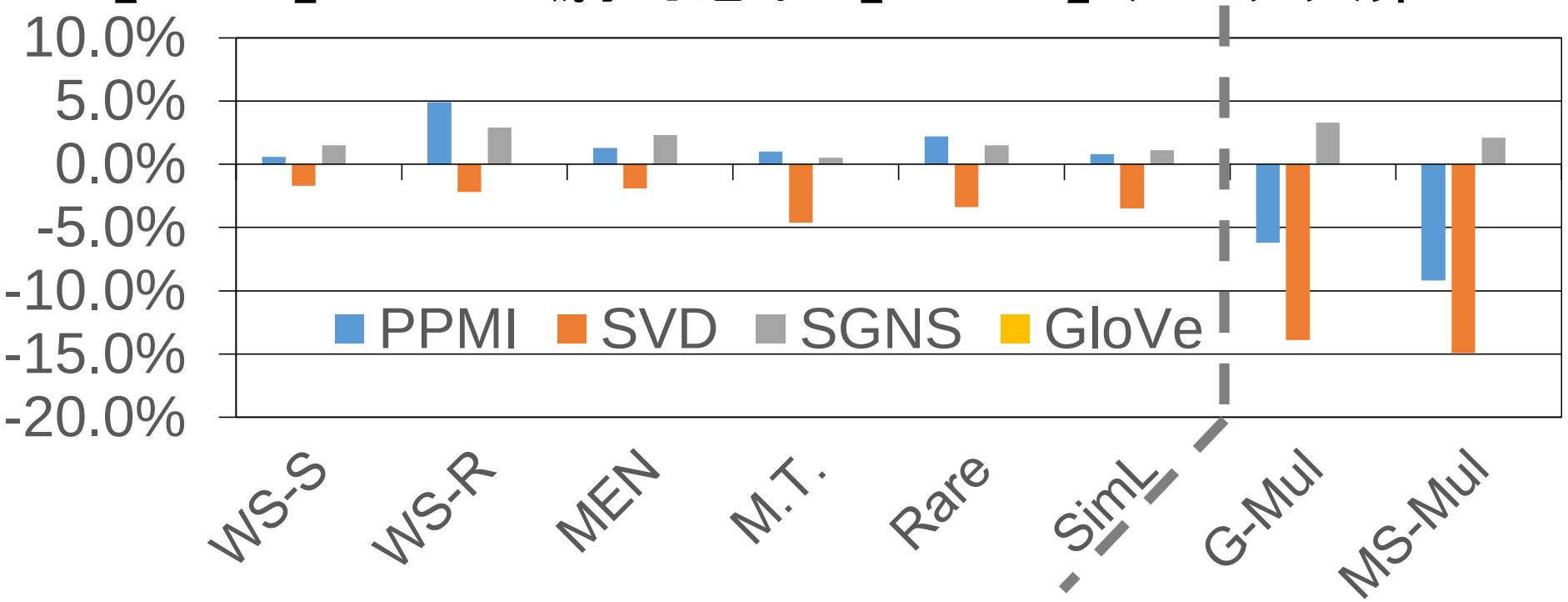
---

* α = 0.75 が良好 (Mikolov et al. 2013)

関連度計算パラメータの効果 (1/2): Shifted PMI

【GloVe】: 設定不可　　【SGNS】: >1が良い

【SVD】:　>1で劇的悪化【PPMI】:タスク次第



↑ **neg** = 1 → > 1 での変化（他パラメータは最適化）

関連度計算パラメータの効果 (1/2): Shifted PMI

【GloVe】: 設定不可　　【SGNS】: >1が良い

【SVD】:　>1で劇的悪化 【PPMI】:タスク次第

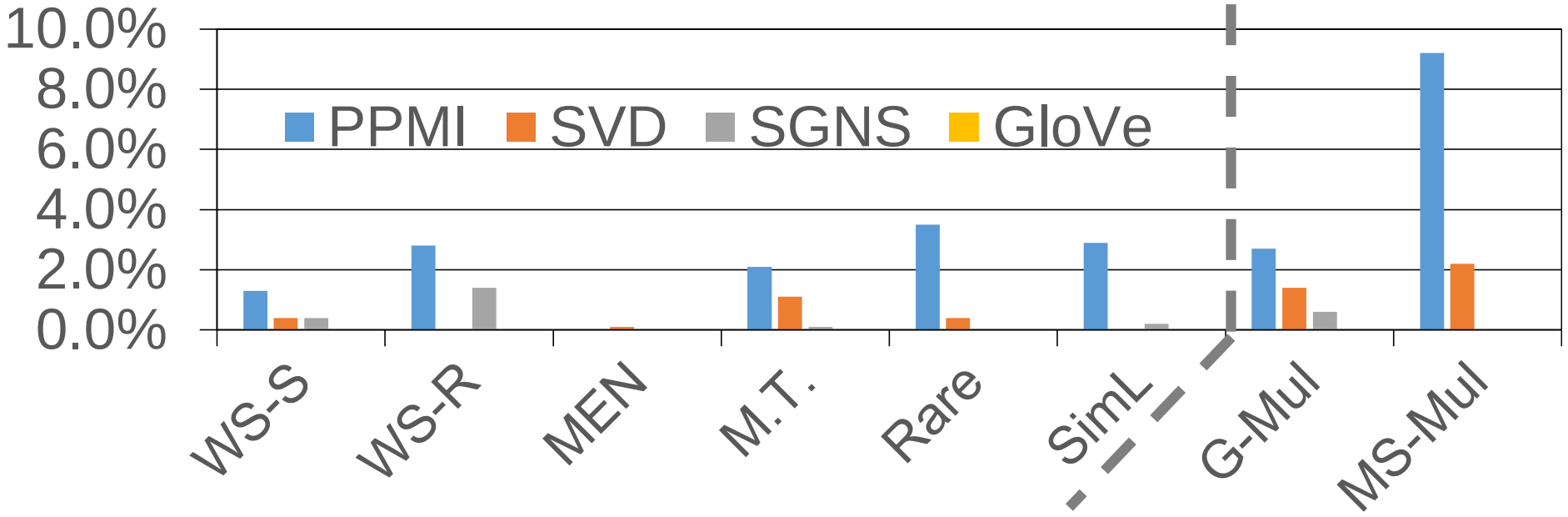
neg→	1	5	15
PPMI	2	6	0
SVD	**8**	0	0
SGNS	0	**4**	**4**
GloVe	-	-	-

← ベスト性能モデルにおける
負例サンプル数（8テスト内訳）

↑ **neg** = 1 → > 1 での変化（他パラメータは最適化）

関連度計算パラメータの効果 (2/2): Context Distribution Smoothing

- どの手法にも有効 (GloVeには適用不可)
  - ◆希な単語の影響を抑えることができる
  - → PMIの弱みを克服できる



↑ **cds** = 1 → 0.75 での変化（他パラメータは最適化）

手法間で共通化された ハイパーパラメータ空間

	パラメータ名	確かめた設定値	導入可能なモデル
前処理	**win**	2, 5, 10	全モデル
	dyn	none, with	全モデル
	sub	none, dirty, clean	全モデル
	del	none, with	全モデル
計算			PPMI, SVD, SGNS
	cd	1, 0.75	PPMI, SVD, SGNS
後処理	**w+c**	only w, $w + c$	SVD, SGNS, GloVe
	eig	0, 0.5, 1	SVD
	nrm	none, row, col, both	全モデル

結果の単語ベクトルを修正

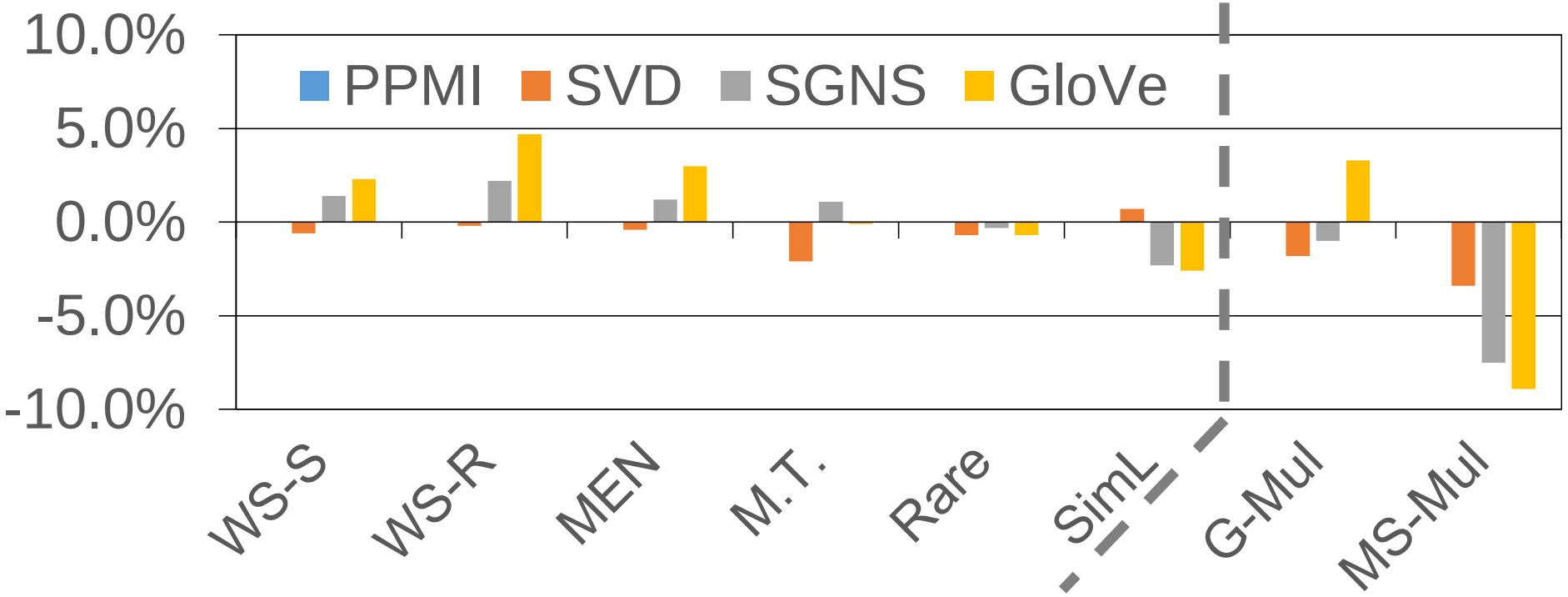
後処理におけるパラメータ: 出力＝単語ベクトルの修正 (1/3)

- **Adding Context Vectors**: **w+c** = [only w/w+c]
  - GloVe: 出力において文脈ベクトルを足す
    - $\vec{v}_{cat} = \vec{w}_{cat} + \vec{c}_{cat}$
  - 単語間 cosine 類似度への効果（詳細割愛）
    - 「2単語が近い文脈で現れやすい」という基準に「お互いの文脈でも現れやすい」という基準を追加
  - 導入方針
    - SVD, SGNS: w, c で異なるベクトル作成 → 可能
    - PPMI: 疎で、追加基準の殆どが無効に → 見送り

後処理パラメータの効果 (1/3): Adding Context Vectors

■（言及はないが… ）効果はまちまち

→（可能なら）事前試行での検証は有意義？



↑ **w+c** = only w → w+c での変化（他パラメータは最適化）

後処理におけるパラメータ: 出力＝単語ベクトルの修正 (2/3)

- **Eigenvalue Weighting: eig** = [0, 0.5, 1]
  - SVDで $W^{SVD} = U_d \cdot \Sigma_d^p$ として p (=**eig**) を調整*

  → W^{SVD}, C^{SVD} に「対称性」を持たせる
    - SGNS の対称性: W^{W2V}, C^{W2V}(非正規直交)
      - 一方に bias かからず → 経験的に良好に働く
    - SVD のデフォルト値 ($p = 1$) では非対称的
      - $W = U_d \cdot \Sigma_d$ (正規直交) $\leftrightarrow$ $C = V_d$ (非正規直交)
    - $p = 0.5$: $W = U_d \cdot \sqrt{\Sigma_d} \leftrightarrow \mathrm{C} = V_d \cdot \sqrt{\Sigma_d}$
    - $p = 0$: $W = U_d \leftrightarrow \mathrm{C} = V_d$

---

* Caron (2001), 他

後処理パラメータの効果 (2/3): Eigenvalue Weighting

- eig = 1: eig=0.5 / 0 と比べて結果が伴わず
  - ◆ 過去研究における SVD 低性能*の原因

  → モチベーションは正しいが、デフォルト (**eig = 1**)のまま利用するのは×

↓ 各 window size での **eig** に伴う性能（他パラメータ最適化）

win	eig	性能
2	0	.612
	0.5	.611
	1	**.551**

win	eig	性能
5	0	.616
	0.5	.612
	1	**.534**

win	eig	性能
10	0	.584
	0.5	.567
	1	**.484**

* Baroni et al. (2014)

性能比較の流れ

【準備】: ハイパーパラメータの共通化

- ◆ ニューラルモデル側のパラメータを明示化
  → 従来モデルでも調整できるよう導入

【実施】: 複数タスクセット上での比較検証

- ◆モデルの（最適パラメータでの）性能比較
- ◆各パラメータの有効性検証

【まとめ】:

- ◆実用上おすすめのモデル・パラメータ選択

まとめ: モデル・パラメータ選択に関する【実用上の】オススメ

- context distribution smoothing (**cds** = 0.75) は PMI の修正として常に使う
  - PPMI, SVD, SGNS にも利用可能
- SVD をデフォルトのまま (**eig** = 1) 使わない → symmetric variants のどれかを使用
- SGNS は頑健なベースライン
  - 必ずしもベストではないがさほど悪くもない
  - 最も学習が早くディスク・メモリを消費せず
- SGNS では多くの負例を利用
- SGNS, GloVe では **w+c** を試す価値あり
  - 再学習の必要もないので試し易い

結論

- よりマイナーと思われがちな最適化部分が実は性能にインパクトがあることを示す
  - 近年のニューラルモデル導入における過度なデザイン偏重に疑問を投げる
- 十分に変数をコントロールした実験が重要
- 透明性のある & 再現可能な実験を
  - 皆コードを公開してはどうでしょうという話
  - この研究はコードを公開
    - http://bitbucket.org/omerlevy/hyperwords

時間の都合でスキップした内容（ご質問あれば紹介いたします…）

- スキップした実験など
  - Analogy タスクにおけるモデル比較
  - ハイパーパラメータ vs. ビッグデータ
  - CBOW との比較
- スキップした分析・データなど
  - GloVe において「context を足す (c+w)」ことが cosine 類似度へ与える効果の解釈
  - 過去の知見に関する検証の補足資料
  - Context Window Size に関する結果詳細

（補足１）
Analogy タスクにおけるモデル比較

実験設定 (2/2):タスクセット（単語類似度／**Analogy**）

■「***a*** is to ***a**** as ***b*** is to ***b**** 」の ***b**** を全語彙から推測する問題セット（2種類）

- ◆ MSR's analogy [Mikolov et a., 2013]
  - ● 8,000問：形態論・統語的な問題設定
  - ●「good is to best as smart is to smartest」
- ◆ Google's analogy [Mikolov et al., 2013]
  - ● 19,544問：統語的5割・意味的問題5割
  - ●「Paris is to France as Tokyo is to Japan」

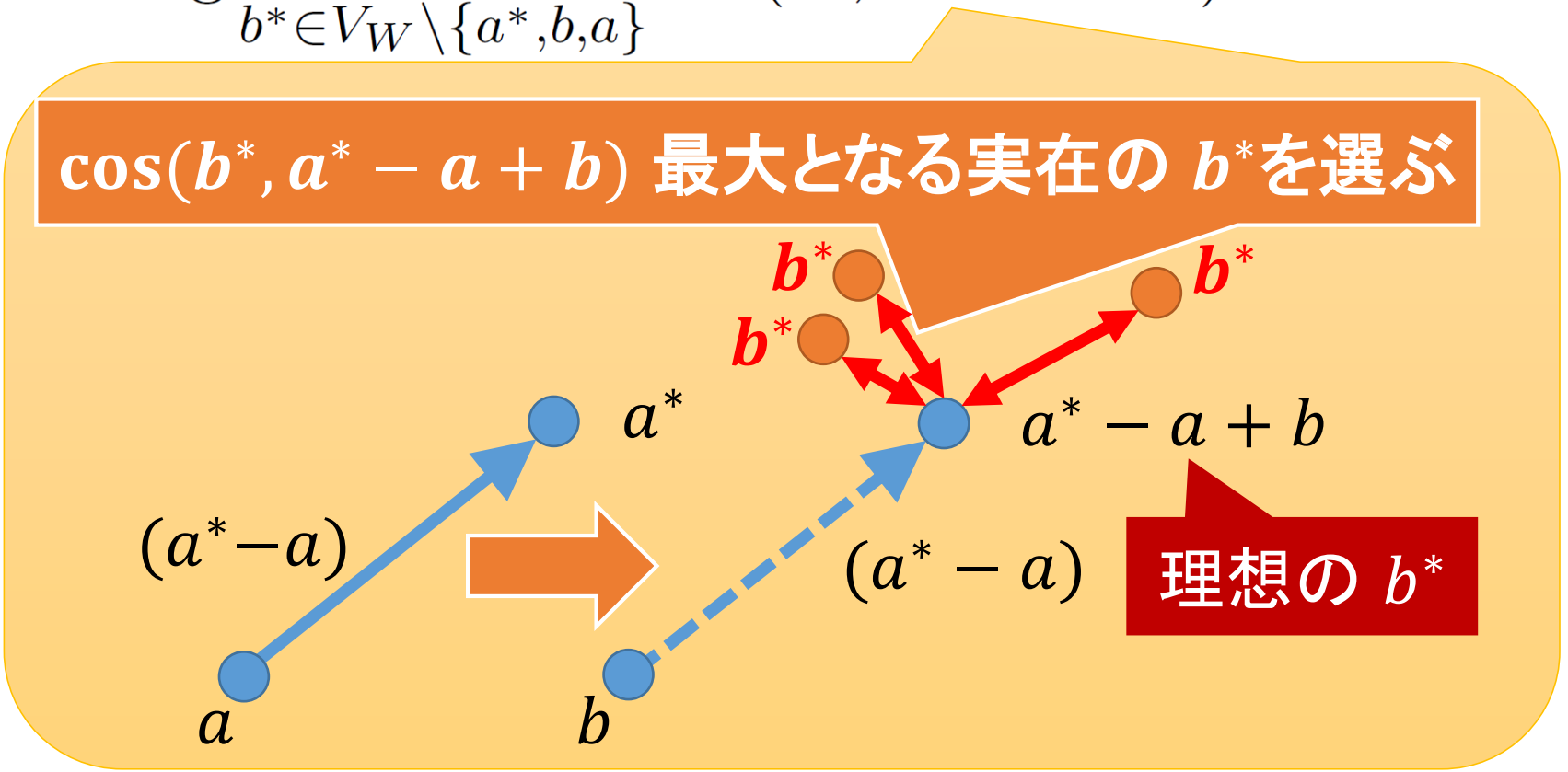
■ Wikipedia 出現100回以上語彙のみに絞る

→ **MSR: 7,118 問／Google: 19,258 問**

Analogy タスクの回答手法

■ 以下の $argmax$ を求める2種方法を実施

$$\arg \max_{b^* \in V_W \setminus \{a^*, b, a\}} \cos(b^*, a^* - a + b)$$



Analogy タスクの回答手法

- 以下の $argmax$ を求める2種方法を実施

$$\arg \max_{b^* \in V_W \setminus \{a^*, b, a\}} \cos(b^*, a^* - a + b)$$

- **3CosAdd** (ベクトル和・差で)

$$\arg \max_{b^* \in V_W \setminus \{a^*, b, a\}} (\cos(b^*, a^*) - \cos(b^*, a) + \cos(b^*, b))$$

- **3CosMul**（乗算で・state-of-the-art*）

$$\arg \max_{b^* \in V_W \setminus \{a^*, b, a\}} \frac{\cos(b^*, a^*) \cdot \cos(b^*, b)}{\cos(b^*, a) + \varepsilon}$$

$\varepsilon = 0.001$
(0除算を回避)

- $argmax$ が正解単語になる割合で評価

* [Levy and Goldberg, 2014]

Analogy タスクの回答手法

- 以下の $argmax$ を求める2種方法を実施

$$\arg \max_{b^* \in V_W \setminus \{a^*, b, a\}} \cos(b^*, a^* - a + b)$$

- **3CosAdd** (ベクトル和・差で)

$$\arg \max_{b^* \in V_W \setminus \{a^*, b, a\}} (\cos(b^*, a) + \cos(b^*, b))$$

データセット名：MS-Add, G-Add

- **3CosMul**（乗算で・state-of-the-art*）

$$\arg \max_{b^* \in V_W \setminus \{a^*, b, a\}} \frac{\cos(b^*, a^*) \cos(b^*, b)}{\cos(b^*, a) + \varepsilon}$$

データセット名：MS-Mul, G-Mul

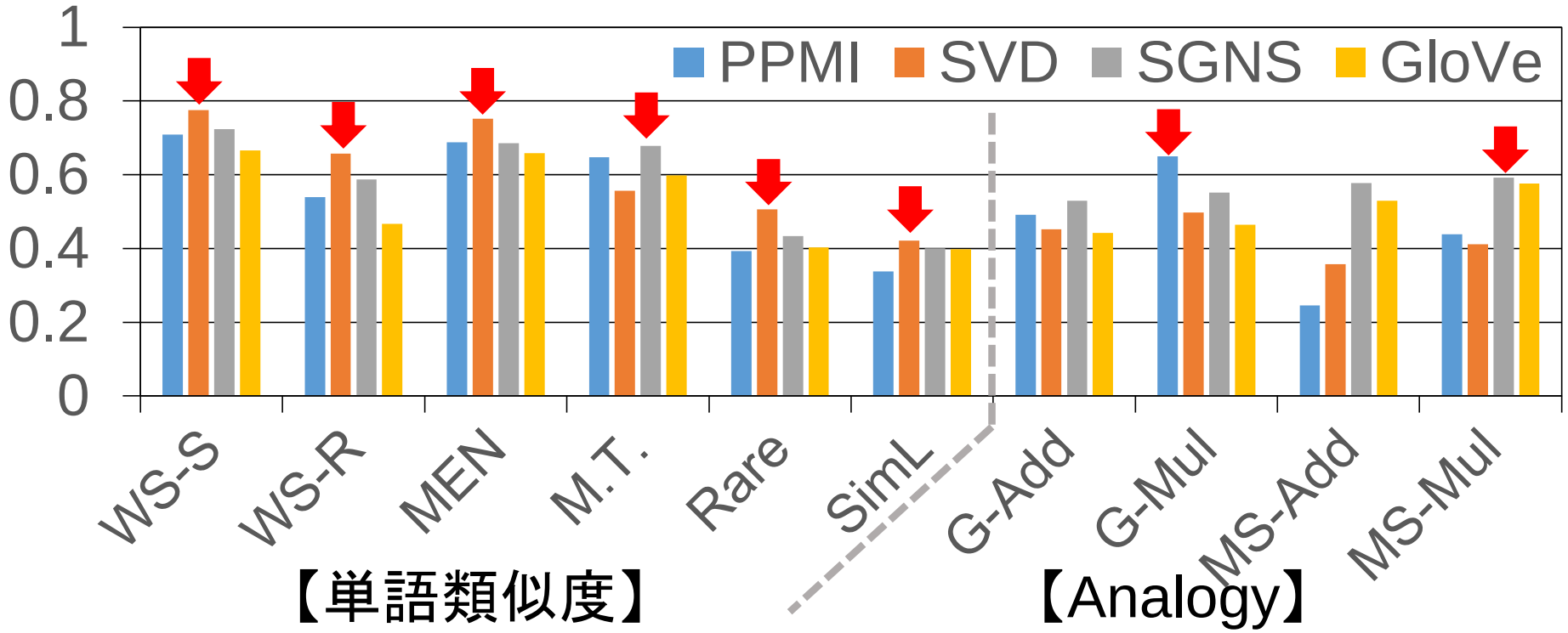
$\varepsilon = 0.001$
(0除算を回避)

- $argmax$ が正解単語になる割合で評価

* [Levy and Goldberg, 2014]

↓ 各データセットでのベスト性能

性能比較 (1/3):
【 頻度ベースモデルに近い設定＊】

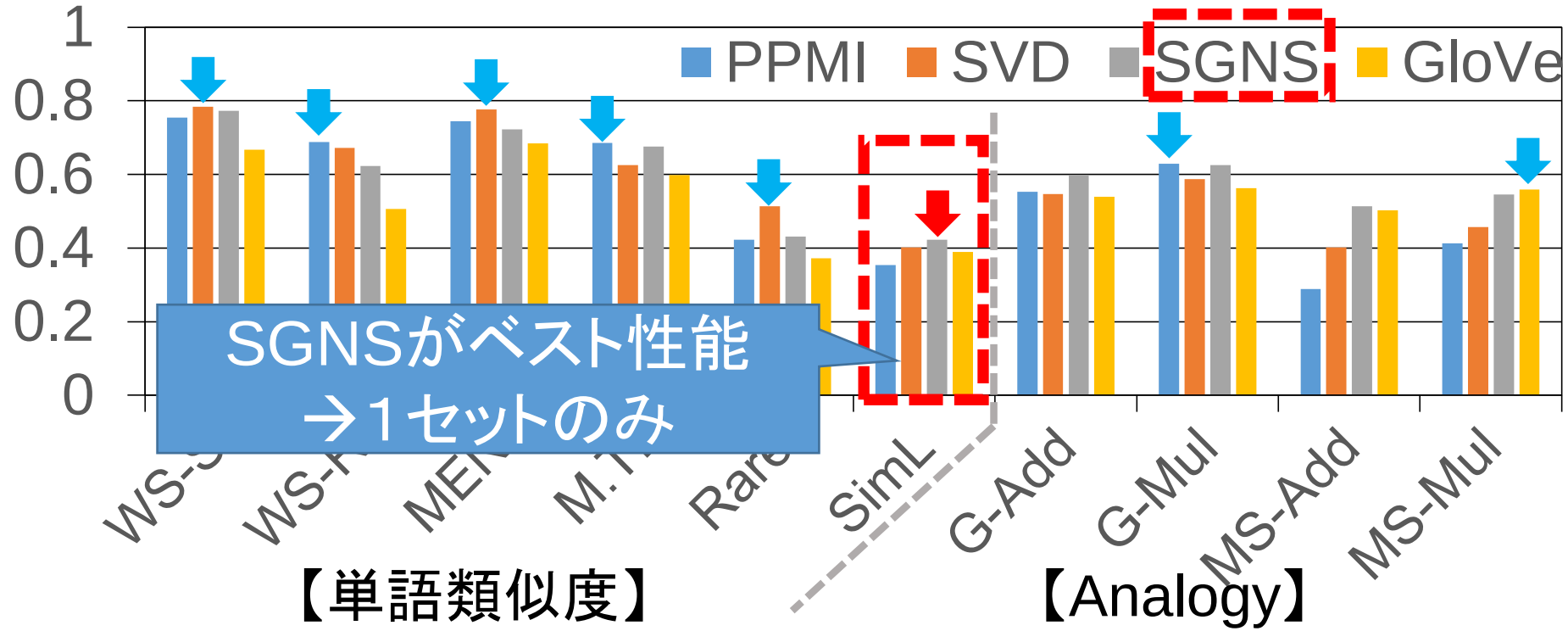


【＊設定値（各詳細は後ほど）】
context window (**win**) = 2
dynamic context (**dyn**) = none
subsampling (**sub**) = none

negative samples (**neg**) = 1
context smoothing (**cds**) = 1
add context vector (**w+c**) = only w
eigenvalue weight (**eig**) = 0.0

⬇ SGNSがベストを達成　⬇ SGNS**以外が**ベストを達成

性能比較 (2/3): 【 word2vec (SGNS) での設定＊】

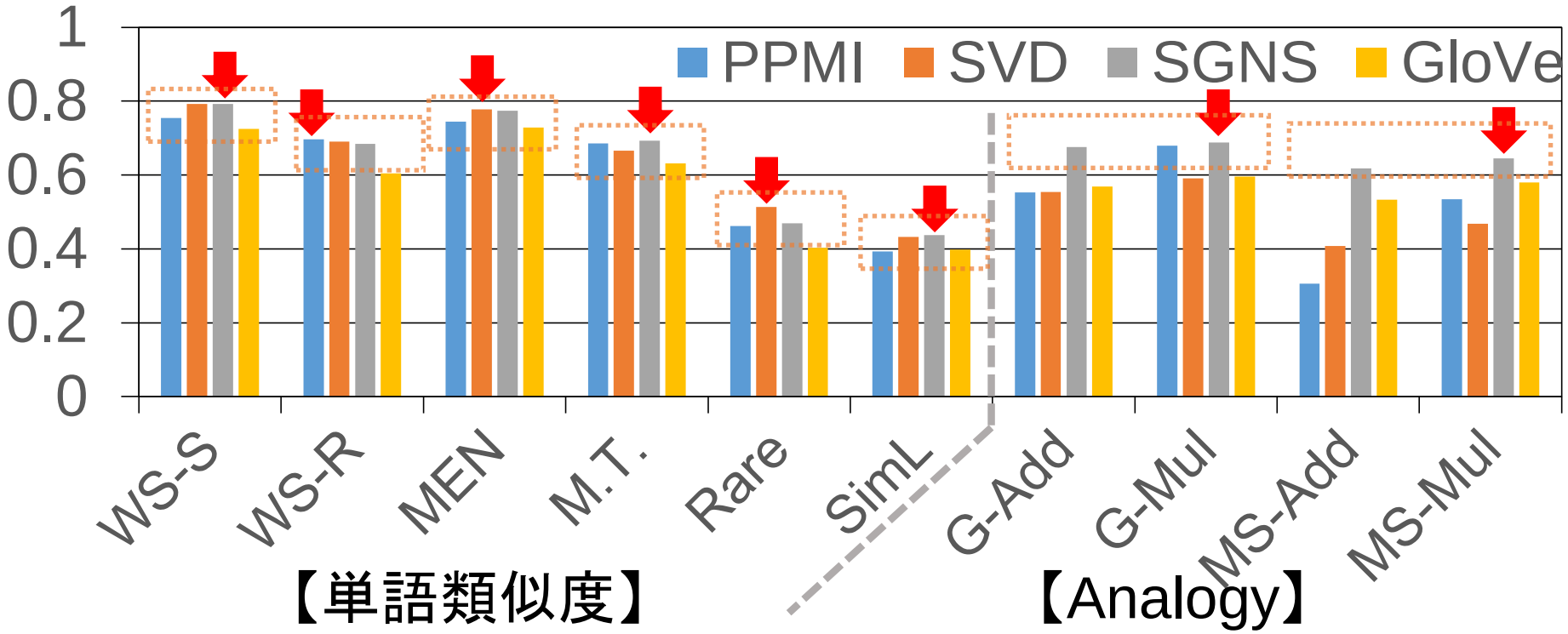


【＊設定値（各詳細は後ほど）】
context window (**win**) = 2
dynamic context (**dyn**) = **with**
subsampling (**sub**) = **dirty**

negative samples (**neg**) = **5**
context smoothing (**cds**) = **0.75**
add context vector (**w+c**) = only *w*
eigenvalue weight (**eig**) = 0.0

各データセットでのベスト性能

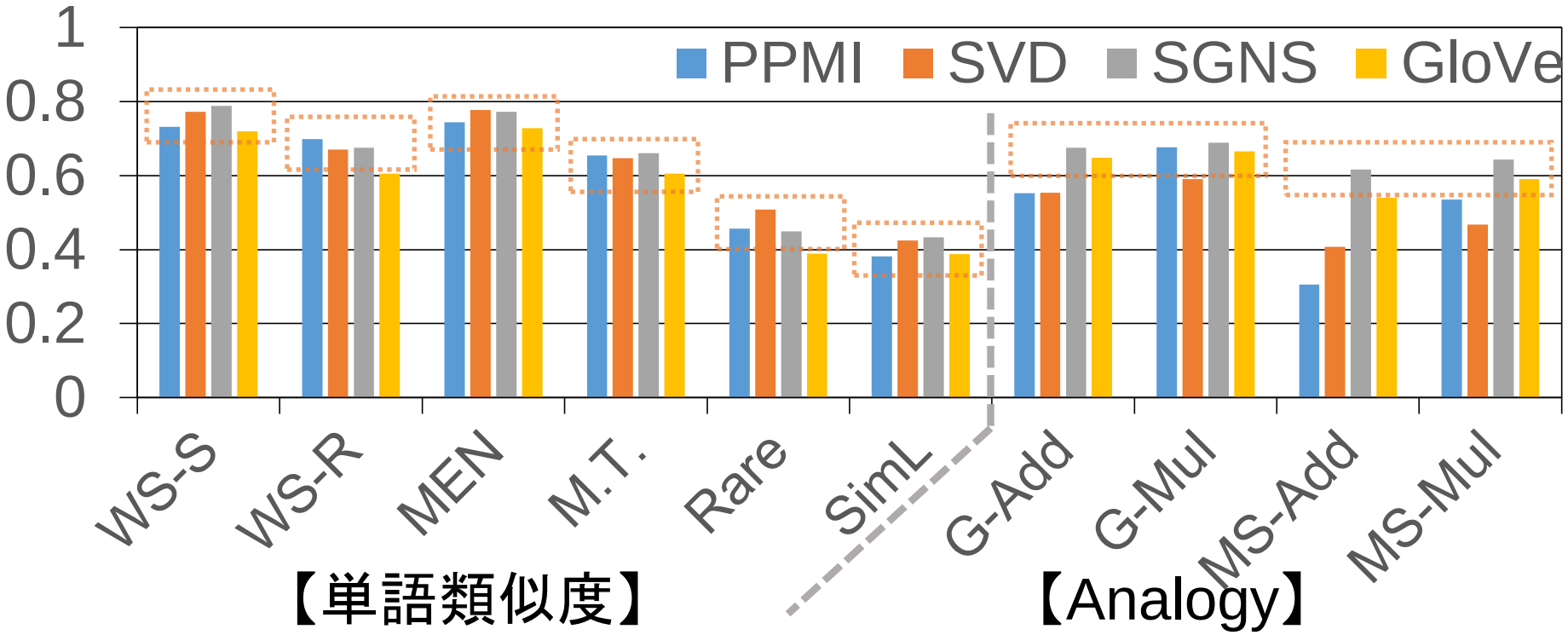
性能比較 (3/3):
【 win = 2 以外をモデル毎に調整】



頻度ベース設定 → word2vec 設定 → 個別調整 で
性能の大幅改善（最大 +0.157, 平均 +0.06）

モデル選択よりパラメータ調整の方がインパクト大

ハイパーパラメータ vs アルゴリズム: (4/4): 学習と評価を分離 (win = 2)



- 同データを用いた2分割交差検定
- 未知データ上評価・学習量半分でも平均 1% 程度の差
- どの手法がベストとは一概に言えない結果

従来の知見は正しいのか？

- **ニューラル ＞ 頻度ベース** *1 → △
  - × 類似度： SGNS平均 ＜ SVD平均
  - ○ Analogy: [SGNS, GloVe] ＞＞ [PPMI, SVD]
- **GloVe ＞ SGNS** *2 → ×
  - ・3CosAdd の1件を除いて全て SGNS ＞ GloVe
- **（analogy において）PPMI ≒ SGNS** *3 → ×
  - ・SGNS の方が (MSR analogy で特に) 優勢
- **3CosMul ＞ 3CosAdd** *3 → ○

*1 Baroni et al., 2014
*2 Pennington et al., 2014
*3 Levy & Goldberg, 2014

（補足２）
ハイパーパラメータ
vs.
ビッグデータ

ハイパーパラメータ vs ビッグデータ (1/3): 実験設定

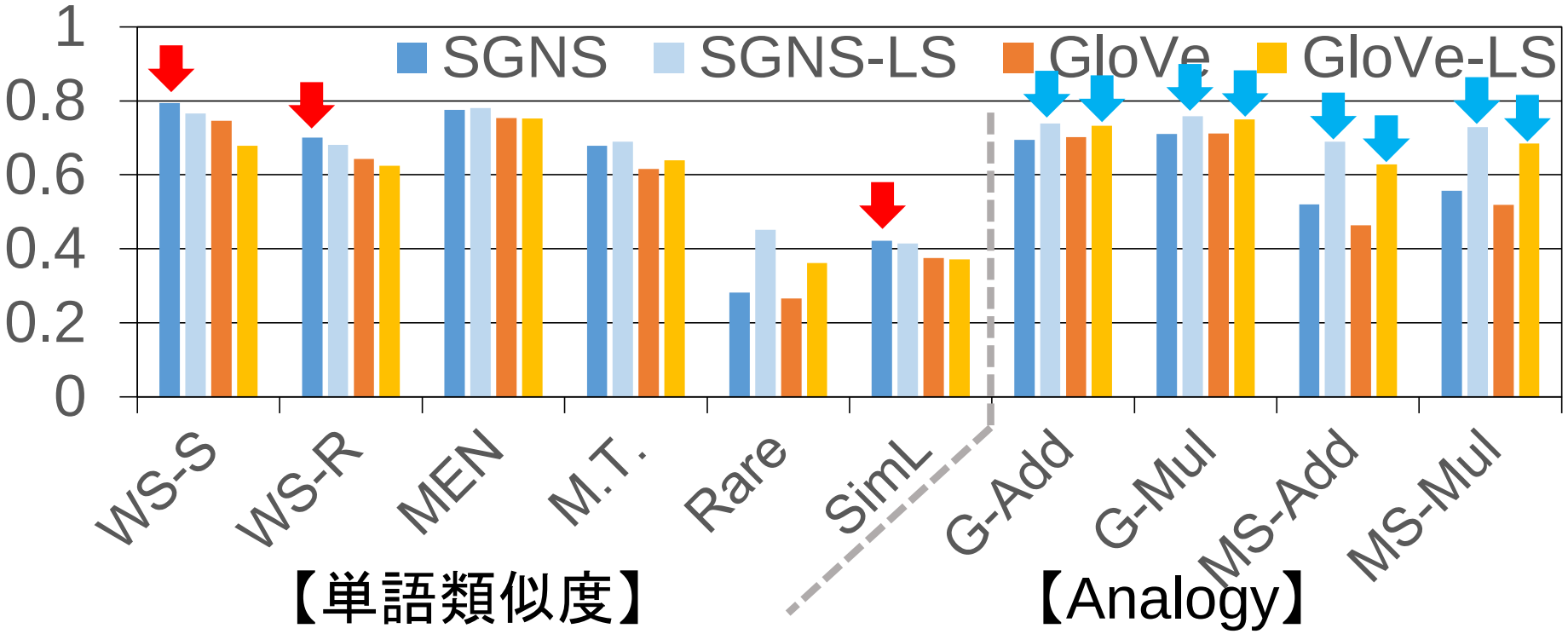
- 105億語超（7倍）からなるコーパスを作成
  - word2vec 用 85億語のコーパス*
    ＋ UKWaC [Ferraresi et al., 2008]
  - 100回以上登場単語だけ使用→ 62万語
- 以下の範囲（$2^4 = 16$通り）で実験
  - broad context window (win = 10)
  - dynamic context window (dyn = with)
  - subsampling (sub = none, dirty)
  - shifted PMI (neg = 1, 5)
  - context distribution smoothing (cds = 1, 0.75)
  - adding context vectors (w+c = only w, w+c)

*word2vec.googlecode.com/svn/trunk/demo-train-big-model-v1.sh

ハイパーパラメータ vs ビッグデータ (2/3): 使用手法の挙動

- SGNS: 半日で完了 (“SGNS-LS”)
- GloVe: 50-iteration を数日で (“GloVe-LS”)
- Count-base の手法をビッグデータで動かすのは技術的に challenging → 実施せず

ハイパーパラメータ vs ビッグデータ (3/3): 実験結果



・単語類似度タスクの 3／6 がコーパス増加よりパラメータ調整の方が効果あり（赤の矢印）
・Analogy にはコーパス増加が有効そう（青の矢印）

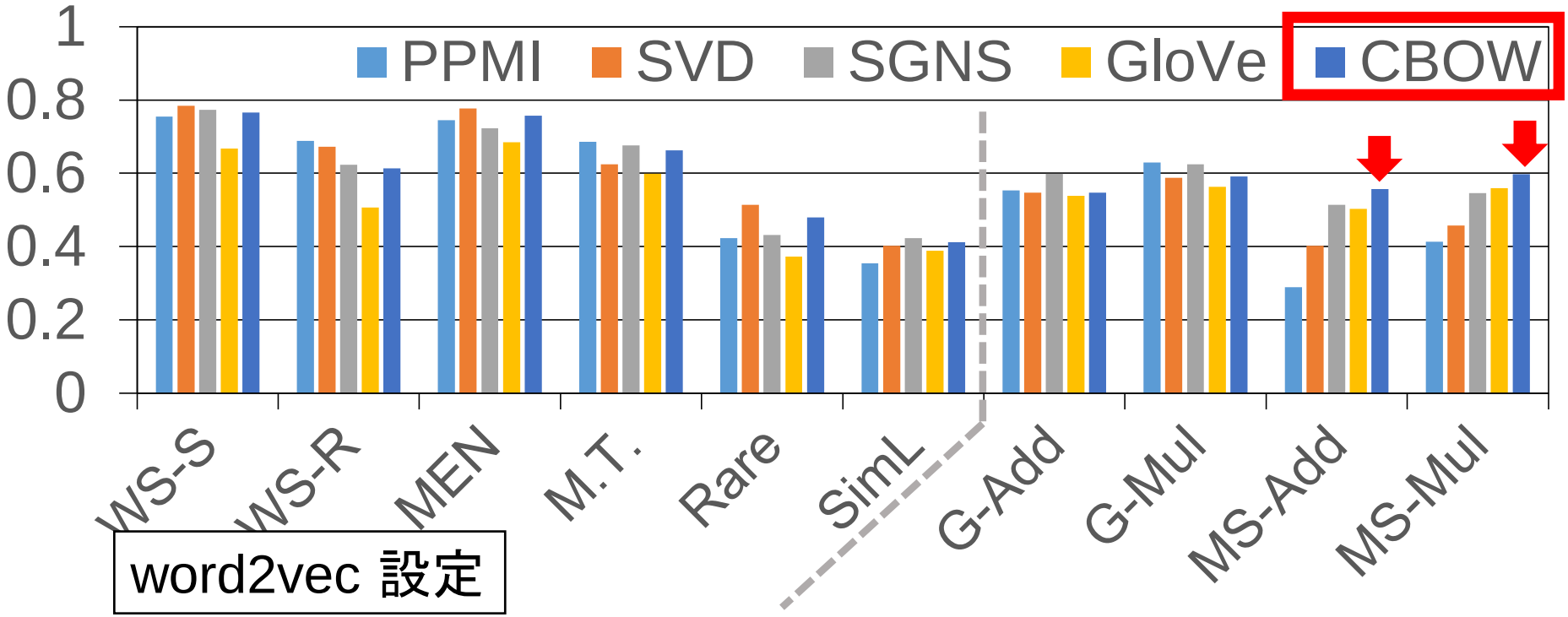
（補足3）
CBOWとの比較

CBOW との比較 (1/2): 背景

- word2vec のもう一つのアルゴリズム
  - context window の各トークンを更に単語ベクトルの和で表現 → more expressive
  - より良い単語表現を引き出せる可能性
- 異なる知見の報告
  - SGNS > CBOW [Mikolov et al. 2013]
  - CBOW > SGNS [Baroni et al. 2014]

CBOW との比較 (2/2): 比較

- MSR Analogy タスクのみベスト
- （単語類似性タスクで改善されたという報告もある [Melamud et al. 2014]）



（補足４）
GloVe において「context を足す (c+w)」ことが cosine類似度へ与える効果の解釈

後処理ハイパーパラメータ (1/3): Adding Context Vectors (w+c)

- GloVe 出力では context ベクトルを足す
  - $\vec{v}_{cat} = \vec{w}_{cat} + \vec{c}_{cat}$
- cosine 類似度への効果（の別解釈）
  ＝ 二次オーダの類似度関数に一次オーダの類似タームを追加

$$cos(x,y) = \frac{\vec{v}_x \cdot \vec{v}_y}{\sqrt{\vec{v}_x \cdot \vec{v}_x}\sqrt{\vec{v}_y \cdot \vec{v}_y}} =$$

$$\frac{(\vec{w}_x + \vec{c}_x) \cdot (\vec{w}_y + \vec{c}_y)}{\sqrt{(\vec{w}_x + \vec{c}_x) \cdot (\vec{w}_x + \vec{c}_x)}\sqrt{(\vec{w}_y + \vec{c}_y) \cdot (\vec{w}_y + \vec{c}_y)}}$$

$$= \frac{\vec{w}_x \cdot \vec{w}_y + \vec{c}_x \cdot \vec{c}_y + \vec{w}_x \cdot \vec{c}_y + \vec{c}_x \cdot \vec{w}_y}{\sqrt{\vec{w}_x^2 + 2\vec{w}_x \cdot \vec{c}_x + \vec{c}_x^2}\sqrt{\vec{w}_y^2 + 2\vec{w}_y \cdot \vec{c}_y + \vec{c}_y^2}}$$

$$= \frac{\vec{w}_x \cdot \vec{w}_y + \vec{c}_x \cdot \vec{c}_y + \vec{w}_x \cdot \vec{c}_y + \vec{c}_x \cdot \vec{w}_y}{2\sqrt{\vec{w}_x \cdot \vec{c}_x + 1}\sqrt{\vec{w}_y \cdot \vec{c}_y + 1}} \quad (4)$$

Word, context ベクトルは正規化されている

後処理ハイパーパラメータ (1/3): Adding Context Vectors (w+c)

- GloVe: 出力では context ベクトルを足す
  - $\vec{v}_{cat} = \vec{w}_{cat} + \vec{c}_{cat}$
- cosine 類似度への効果 ＝二次オーダの類似度関数に一次オーダの類似項を追加

$$cos(x, y) = \frac{\vec{w}_x \cdot \vec{w}_y + \vec{c}_x \cdot \vec{c}_y + \vec{w}_x \cdot \vec{c}_y + \vec{c}_x \cdot \vec{w}_y}{2\sqrt{\vec{w}_x \cdot \vec{c}_x + 1}\sqrt{\vec{w}_y \cdot \vec{c}_y + 1}}$$

後処理ハイパーパラメータ (1/3): Adding Context Vectors (w+c)

- GloVe: 出力では context ベクトルを足す
  - $\vec{v}_{cat} = \vec{w}_{cat} + \vec{c}_{cat}$
- cosine 類似度への効果 ＝二次オーダの類似度関数に一次オーダの類似項を追加

2単語が似た context で出現傾向に基づき「置換可能」か否か

片方の単語がもう一方の単語の context で出現する傾向
[SVD, SGNS → PMI(w,c) ／ GloVe → log-count (+bias)]

$$cos(x,y) = \frac{\vec{w}_x \cdot \vec{w}_y + \vec{c}_x \cdot \vec{c}_y + \vec{w}_x \cdot \vec{c}_y + \vec{c}_x \cdot \vec{w}_y}{2\sqrt{\vec{w}_x \cdot \vec{c}_x + 1}\sqrt{\vec{w}_y \cdot \vec{c}_y + 1}}$$

後処理ハイパーパラメータ (1/3): Adding Context Vectors (w+c)

- GloVe: 出力では context ベクトルを足す
  - $\vec{v}_{cat} = \vec{w}_{cat} + \vec{c}_{cat}$
- cosine 類似度への効果 ＝二次オーダの類似度関数に一次オーダの類似項を追加

$$sim(x,y) = \frac{sim_2(x,y) + sim_1(x,y)}{\sqrt{sim_1(x,x)+1}\sqrt{sim_1(y,y)+1}}$$

- 2単語が近い
  ＝「2単語が近いcontext で現れやすい」
  or「お互いの context で現れやすい」

後処理ハイパーパラメータ (1/3): Adding Context Vectors (w+c)

- GloVe: 出力では context ベクトルを足す
  - $\vec{v}_{cat} = \vec{w}_{cat} + \vec{c}_{cat}$
- cosine 類似度への効果 ＝二次オーダの類似度関数に一次オーダの類似項を追加
- 導入方針
  - SVD, SGNS: w, c で異なるベクトル作成
    → trivial に導入可能
  - PPMI: 疎で、一次類似の殆どが無効
    → 今回は導入を見送る
- **w+c = [only w / w+c]**

（補足５）
過去の知見に関する検証の補足資料

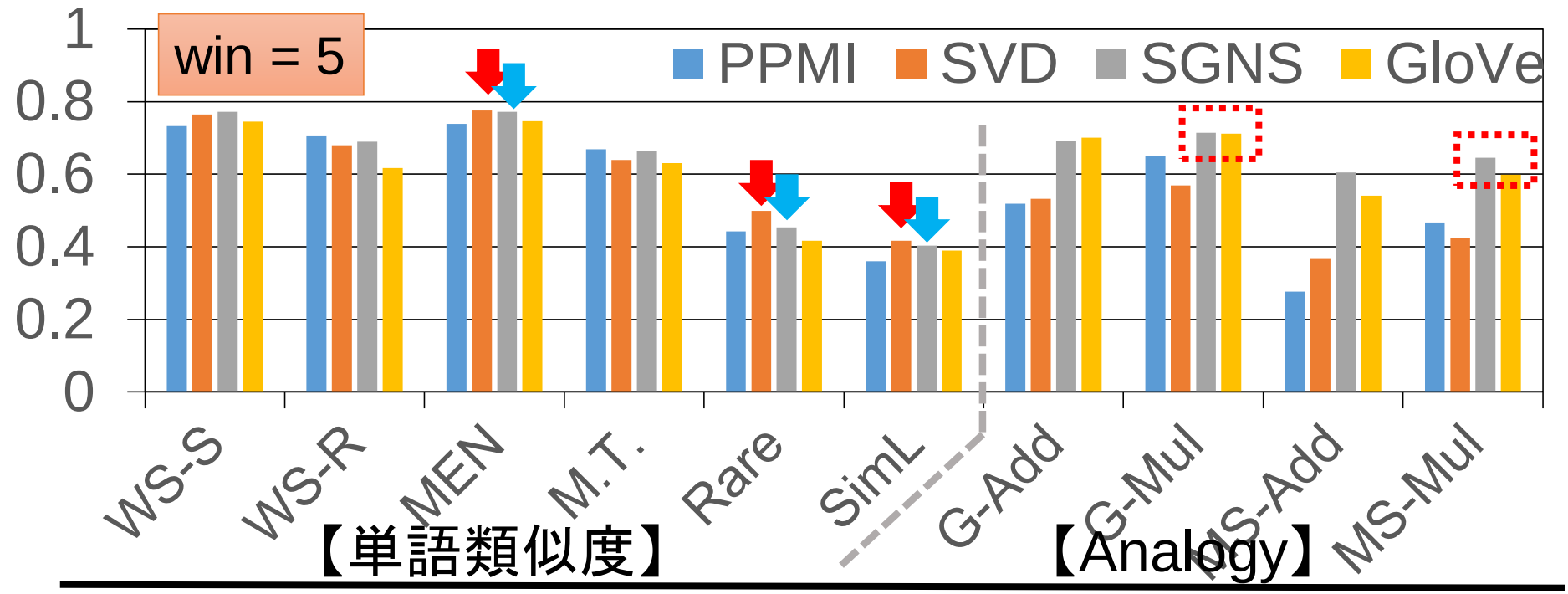
従来の知見は正しいのか？

- **ニューラル ＞ 頻度ベース** *1 → △
  - × 類似度： SGNS平均 ＜ SVD平均
  - ○ Analogy: [SGNS, GloVe] ＞＞ [PPMI, SVD]
- **GloVe ＞ SGNS** *2 → ×
  - ・3CosAdd の1件を除いて全て SGNS ＞ GloVe
- **（analogy において）PPMI ≒ SGNS** *3 → ×
  - ・SGNS の方が (MSR analogy で特に) 優勢
- **3CosMul ＞ 3CosAdd** *3 → ○

*1 Baroni et al., 2014
*2 Pennington et al., 2014
*3 Levy & Goldberg, 2014

過去の主張の再評価 (1/4): embedding は優れている*のか？

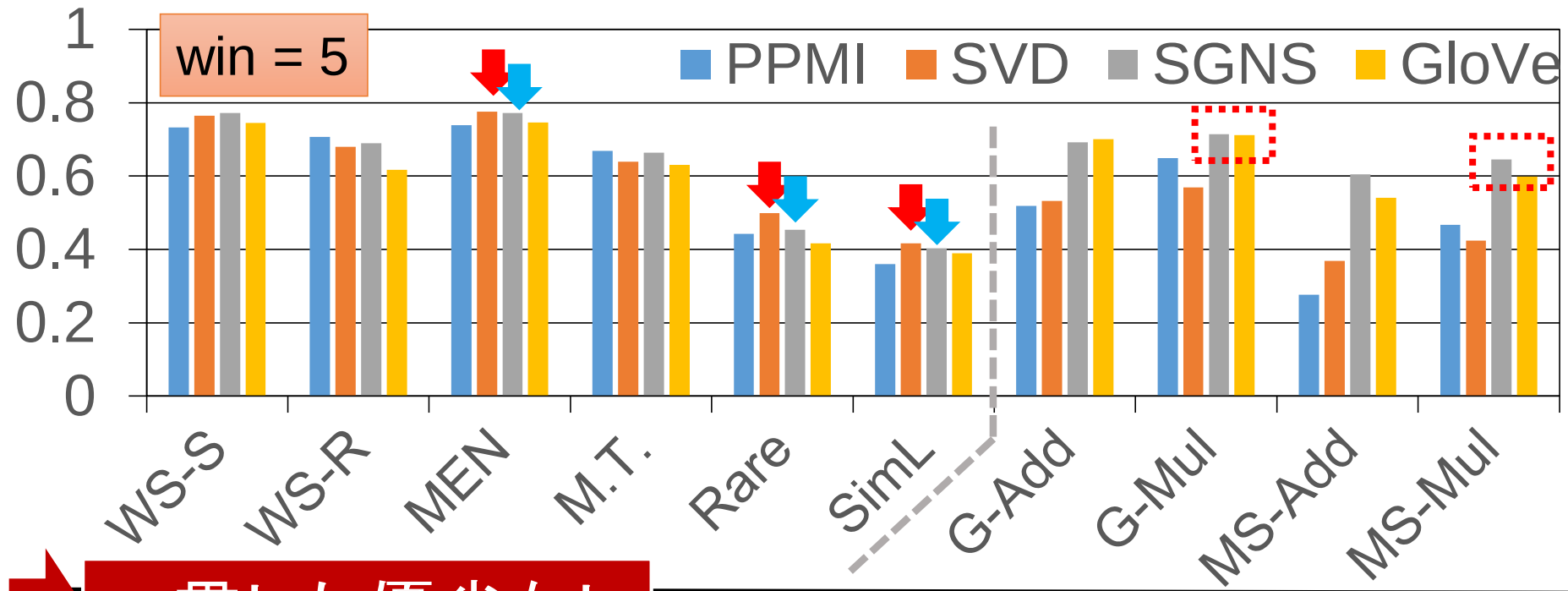
- 類似度:　[SGNS平均] < [SVD平均] (win=2,5)
- analogy: [SGNS, GloVe] >> [PPMI, SVD]



* Baroni et al., 2014 による一連の評価

過去の主張の再評価 (1/4): embedding は優れている*のか？

- 類似度:　[SGNS平均] < [SVD平均] (win=2,5)
- analogy: [SGNS, GloVe] >> [PPMI, SVD]



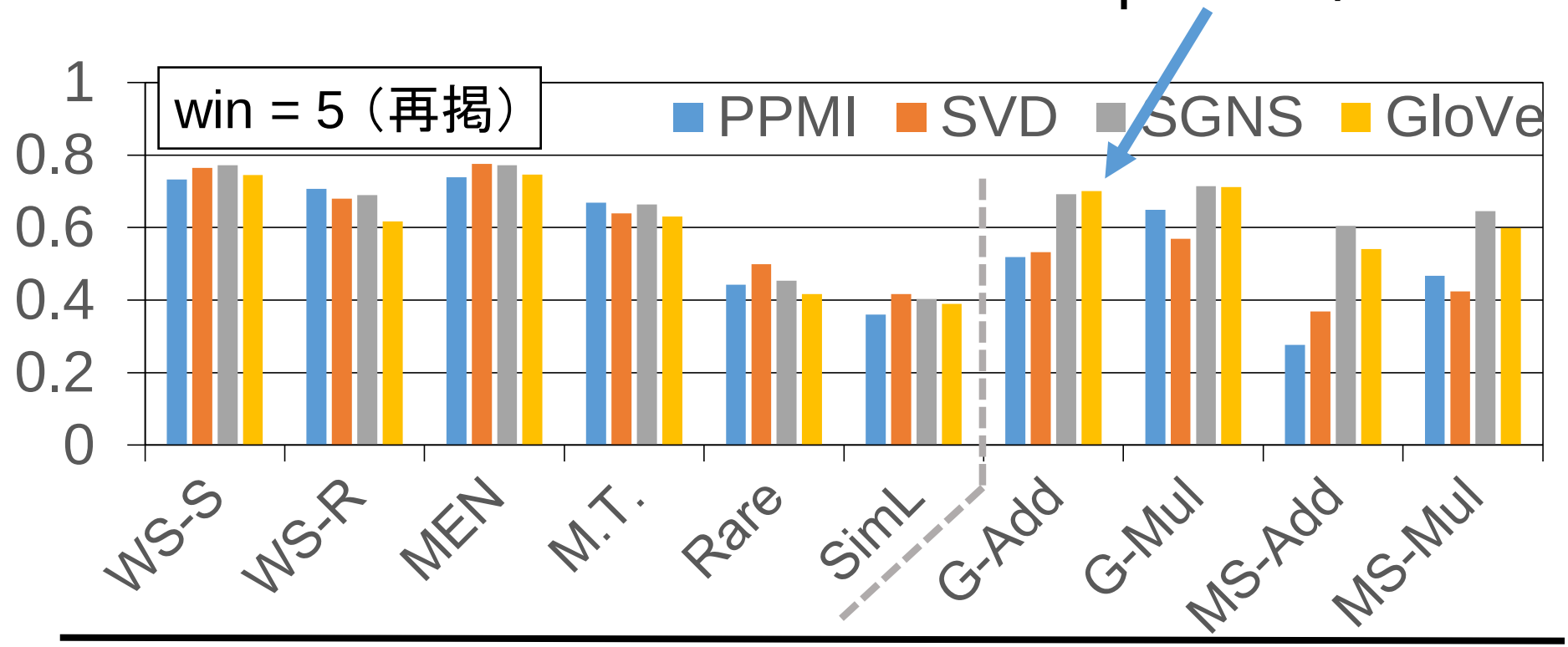
一貫した優劣なし

* Baroni et al., 2014 による一連の評価

過去の主張の再評価 (2/4): GloVe はSGNSより優れている*のか？

■ あらゆるタスクで SGNS > GloVe

◆ 3CosAdd のみ GloVe が 0.8point上回る



* Pennington et al., 2014 による一連の実証

過去の主張の再評価 (2/4): GloVe はSGNSより優れている*のか？

- あらゆるタスクで SGNS > GloVe
- 過去実験*との設定の違い
  - ハイパーパラメータが可変（特にw+c）
  - Google analogy 以外でも評価している
  - 全手法を同じコーパス上で比較している
- GloVe では shifted PMI(neg) と context distribution smoothing (cds)が利用不可
  - 代わりに、より潜在性のある bias 調整ができるのに SGNS と決定的な差がつかず

---

* Pennington et al., 2014 による一連の実証

過去の主張の再評価 (2/4): GloVe はSGNSより優れている*のか？

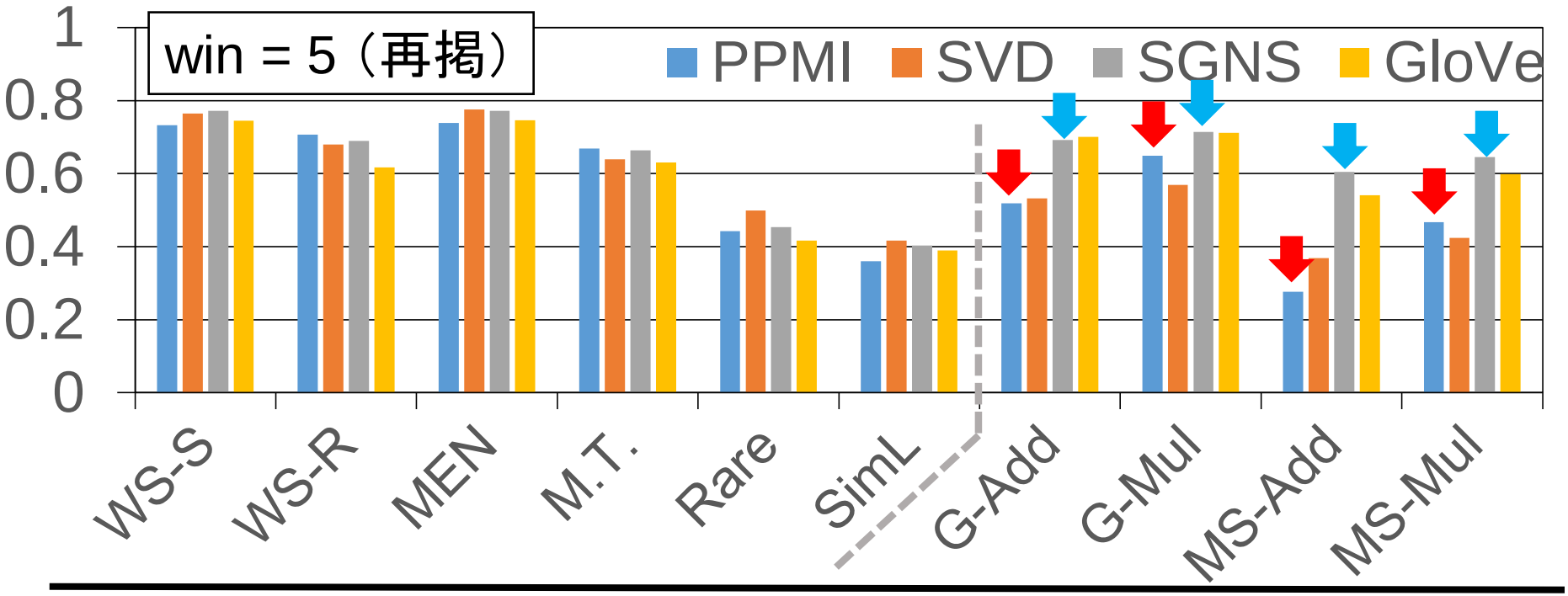
- あらゆるタスクで SGNS > GloVe
- 過去実験*との設定の違い
  - ハイパーパラメータが可変（特にw+c）
  - Google analogy 以外でも評価している
  - 全手法を同じコーパス上で比較している
- GloVe では shifted PMI(neg) と context distribution smoothing (cds)が利用不可
  - 代わりに、より潜在性のある bias 調整ができるのに SGNS と決定的な差がつかず

SGNS の方が優れているように見える

:証

過去の主張の再評価 (3/4): analogy タスクにおいてPPMIはSGNSと同等*か？

- SGNS の方が明らかに優勢 (⬇)
  - MSR analogy タスクにおいて特に顕著



* Levy & Goldberg, 2014 (Google/MSR analogy タスクでの検証)

過去の主張の再評価 (3/4): analogy タスクにおいてPPMIはSGNSと同等*か？

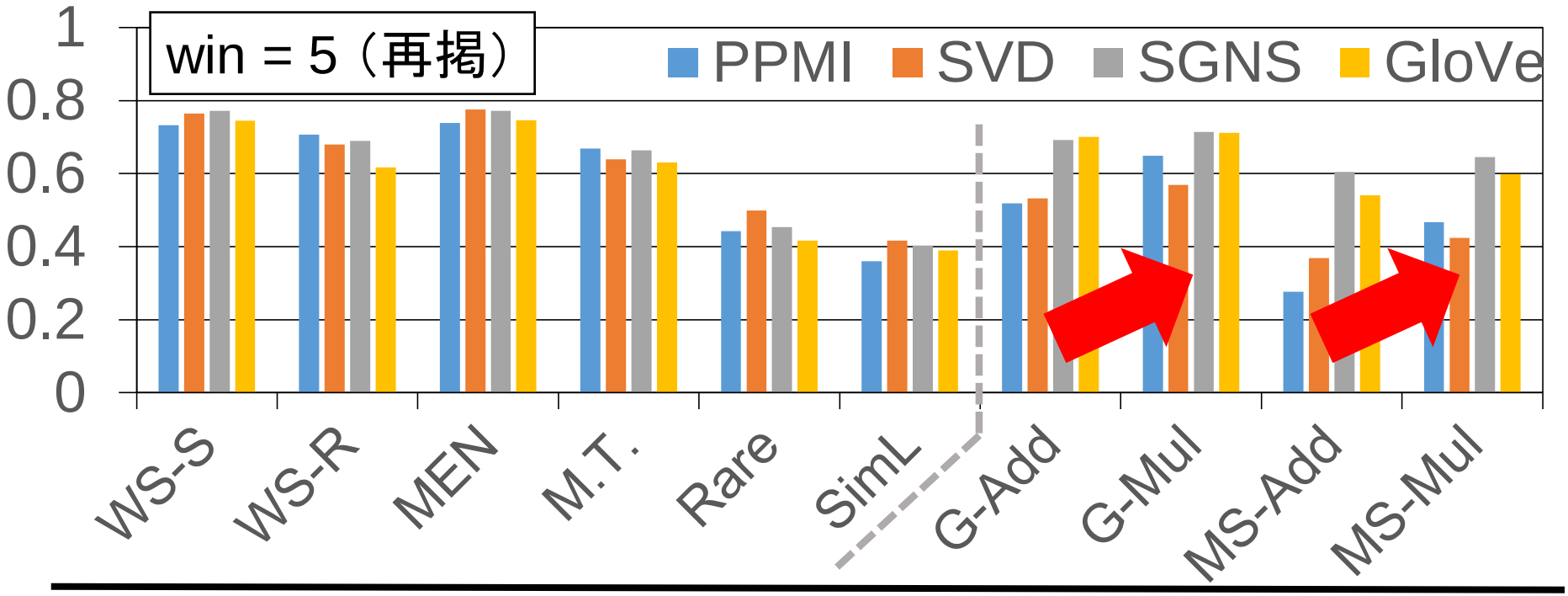
- SGNS の方が明らかに優勢
  - MSR analogy タスクにおいて特に顕著
- MSR→統語的関係（単複・時制変化）
  - context の把握（冠詞・機能語）が重要か
    - SGNS は個々の例への重みの付け方／(PPMIではfilterされる)負の相関等でうまく把握可能？
    - PPMI で単語に相対位置情報を付けることにより、比較的良い成績が確認されている*

---

* Levy & Goldberg, 2014 (Google/MSR analogy タスクでの検証)

過去の主張の再評価 (4/4): 3CosMul は 3CosAdd より多くの analogy を再現可能*か？

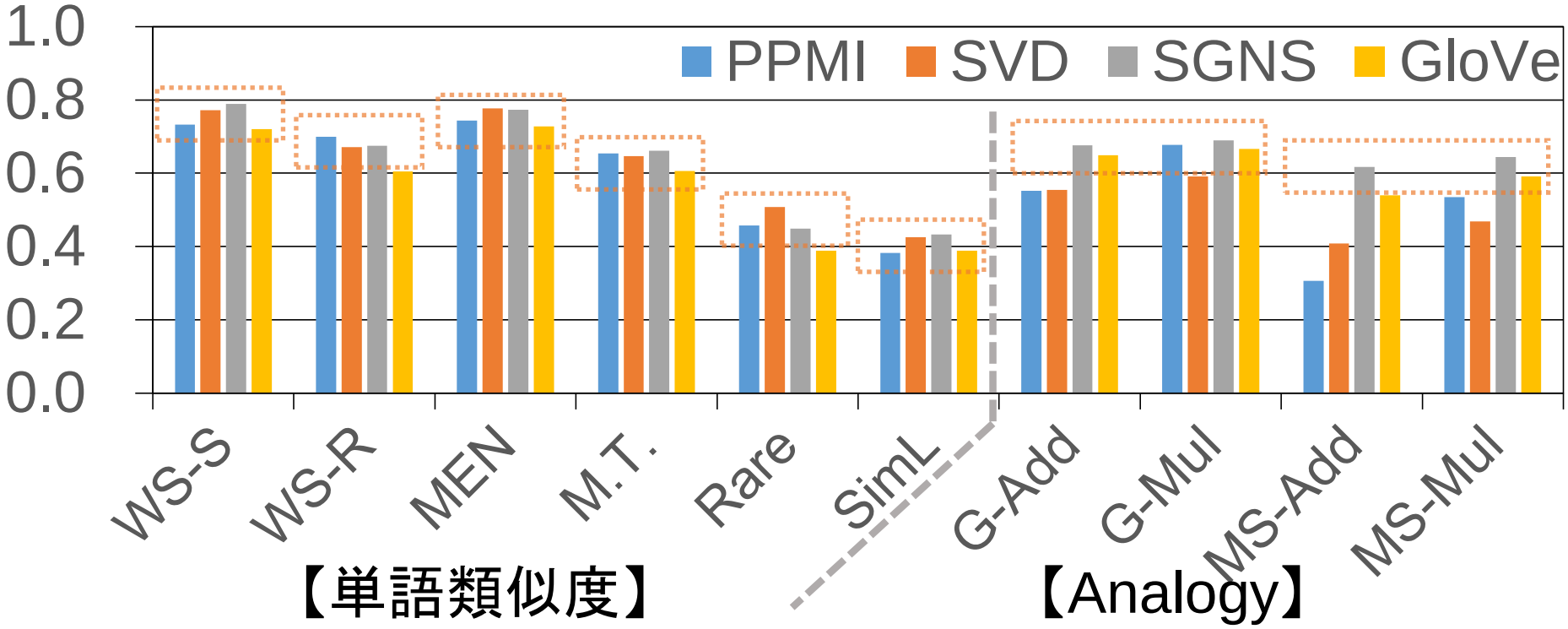
- 過去の知見と今回の知見が一致
- SVD と PPMI で特に顕著



* Levy & Goldberg, 2014 (Google/MSR analogy タスクでの検証)

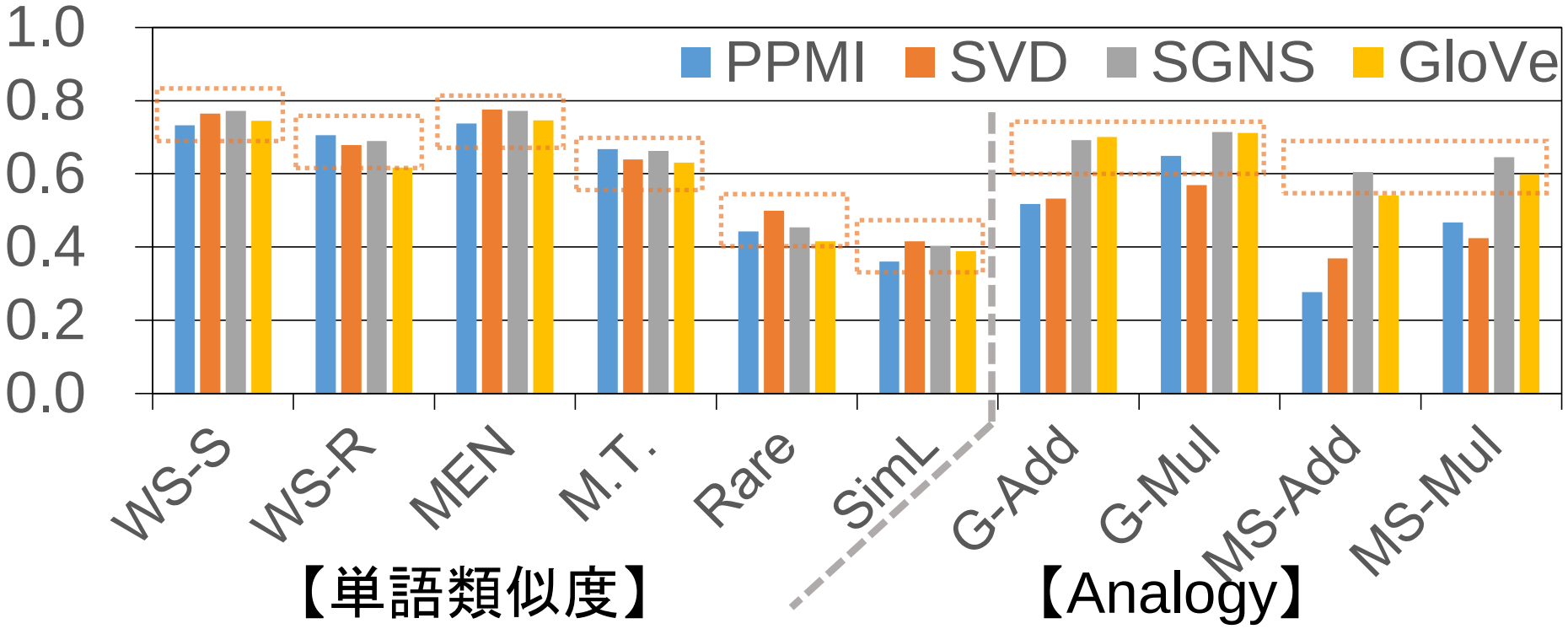
（補足６）
Context Window Size に関する詳細結果

ハイパーパラメータ vs アルゴリズム: (4/6): 学習と評価を分離 (**win = 2**)



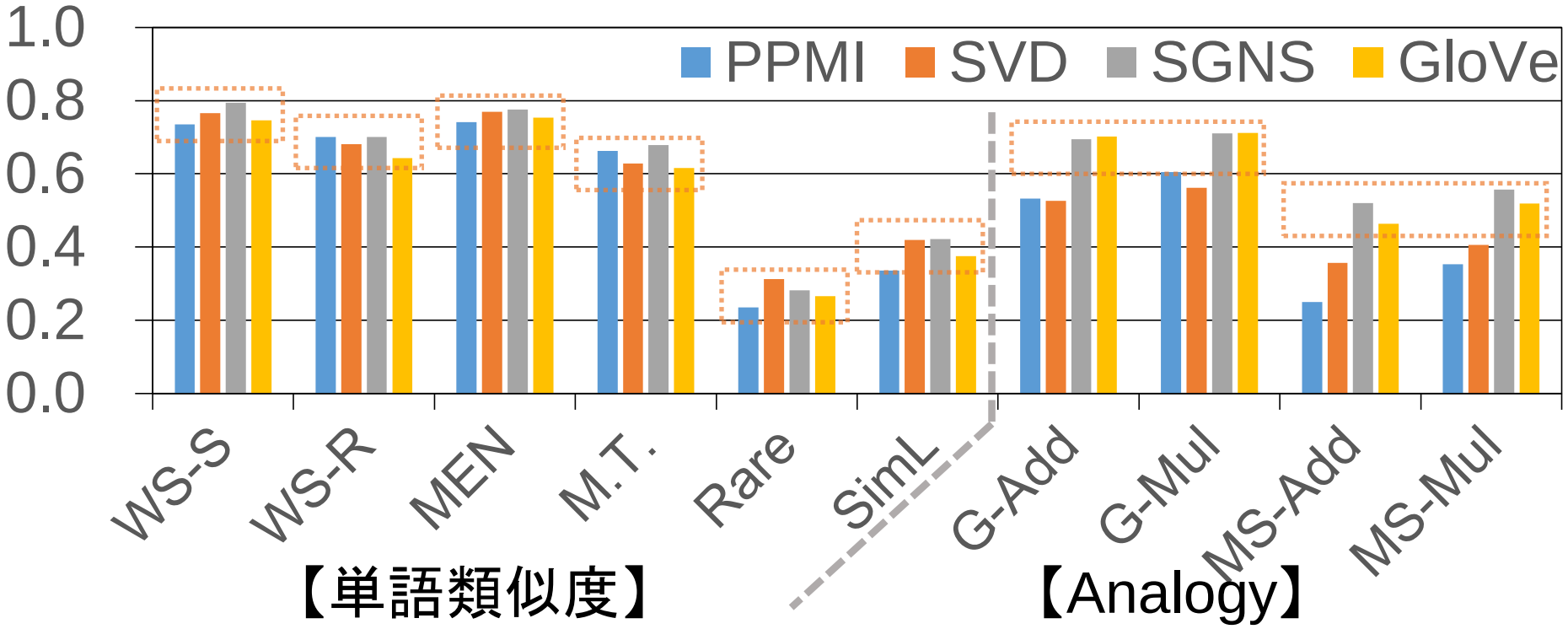
- 同データを用いた2分割交差検定
- 未知データ上評価・学習量半分でも平均 1% 程度の差
- どの手法がベストとは一概に言えない結果

ハイパーパラメータ vs アルゴリズム: (5/6): 学習と評価を分離 (**win = 5**)



- 同データを用いた2分割交差検定
- 未知データ上評価・学習量半分でも平均 1% 程度の差
- どの手法がベストとは一概に言えない結果

ハイパーパラメータ vs アルゴリズム: (6/6): 学習と評価を分離 (**win = 10**)



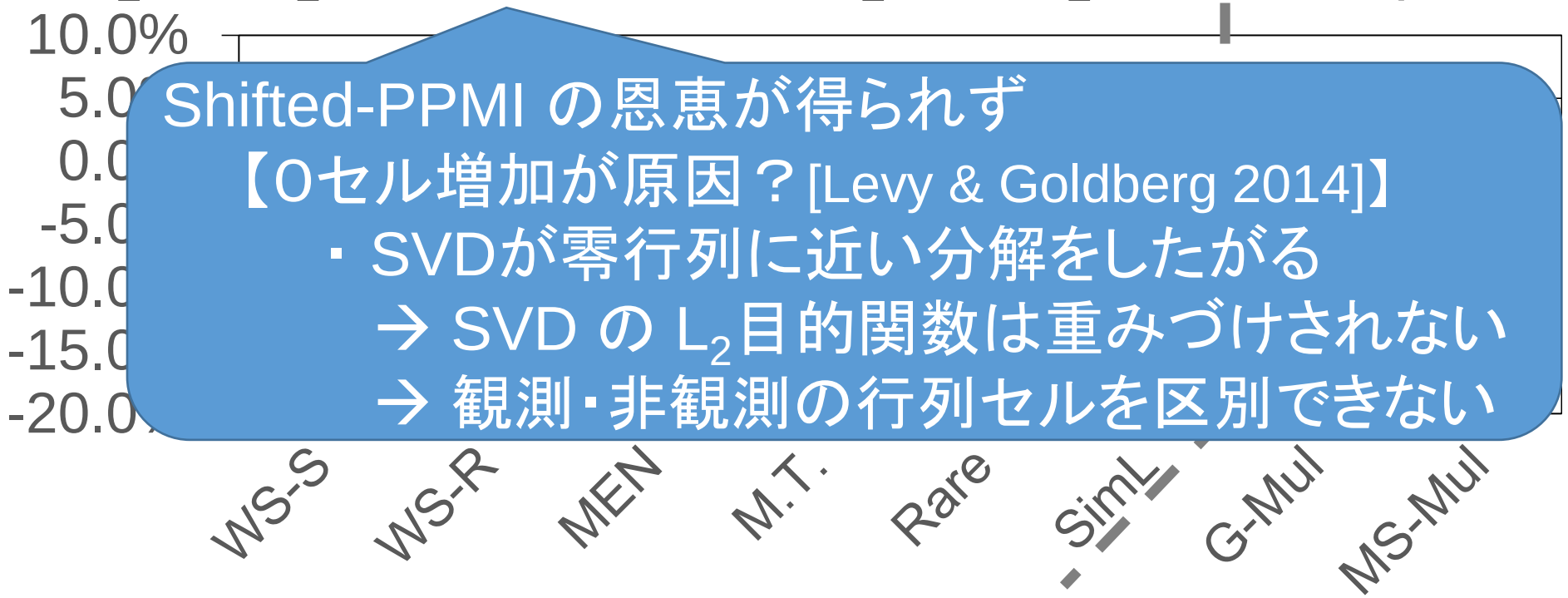
- 同データを用いた2分割交差検定
- 未知データ上評価・学習量半分でも平均 1% 程度の差
- どの手法がベストとは一概に言えない結果

（補足7）
その他諸々
（スライドへの追記など）

関連度計算パラメータの効果 (1/2): Shifted PMI

【GloVe】: 設定不可　　【SGNS】: >1が良い

【SVD】:　>1で劇的悪化【PPMI】:タスク次第



Shifted-PPMI の恩恵が得られず

【0セル増加が原因？[Levy & Goldberg 2014】

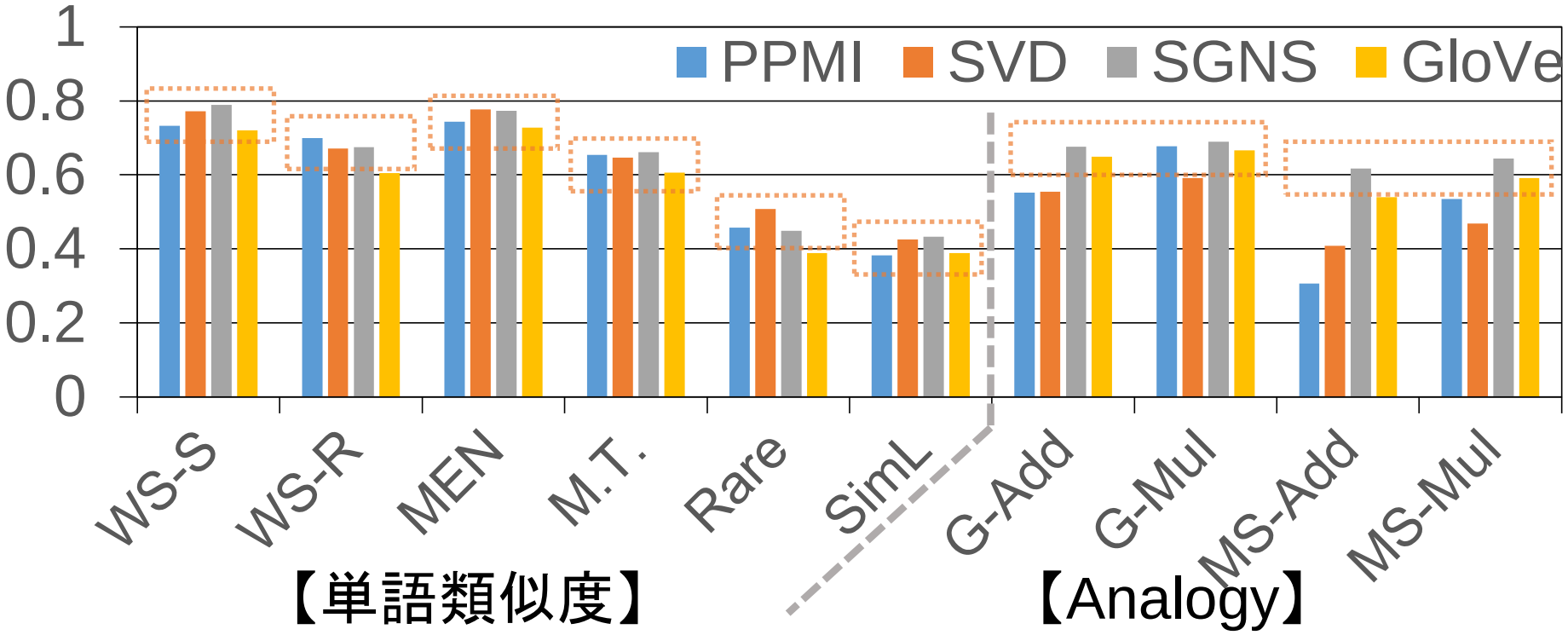
・SVDが零行列に近い分解をしたがる

→ SVD の L_2目的関数は重みづけされない

→ 観測・非観測の行列セルを区別できない

↑ **neg** = 1 → > 1 での変化（他パラメータは最適化）

性能比較 (4/4):
【学習・評価セットを分離】



- 同データを用いた2分割交差検定
- 未知データ上評価・学習量半分でも平均 1% 程度の差
- どの手法がベストとは一概に言えない結果

（事前実験により効果がなかったため固定されたパラメータ1）

前処理における共通化：SGNS のパラメータ → 全手法へ

- **Dynamic context window**：**dyn** = [none/with]
  - 各トークン毎に context を 1~window size (**win**) でサンプリング→ 距離に基づく重みを動的に決定
- **Subsampling**: **sub** = [none/dirty/clean]
  - 頻度f ≥閾値tの単語を確率$1-\sqrt{t/f}$で除去*
  - context 取得前(dirty)・後(clean)実施で性能差は見られず → $t = 10^{-5}$, dirty（word2vec 準拠）
- **Deleting Rare Words**: **del** = [none/with]
  - 希単語の除去（事前実験で効果なし→カット）

*word2vec では確率 $(f-t)/f-\sqrt{t/f}$ で除去

（事前実験により効果がなかったため固定されたパラメータ2）

後処理におけるパラメータ: 出力＝単語ベクトルの修正 (3/3)

■ **Vector Normalization:**

nrm = [row / none / column / both]

- row: 上記の（行）正規化
- none: 全く正規化しない
- column: W の行ではなく列を正規化*
  [Pennington et al. 2014]
- both: 列も行も正規化

→ 事前実験により **nrm = [row]** で固定

---

* SVD で固有値を捨てる (eig=0) → SVD では効果あり