OLYMPUS

INTERNAL USE ONLY

word2vecの紹介(2)

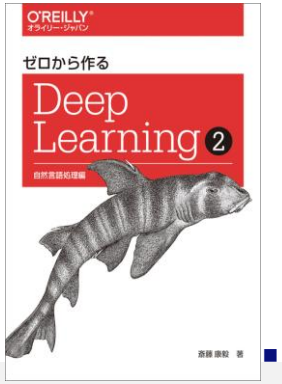
自然言語処理のための単語のベクトル化

100 YEARS
Endeavor for Better

科学ソフト開発 | 秋山光弘 | 2020年7月15日

以前の発表では、以下の参考書籍に従い、word2vecというテキストデータから単語ベクトル群(辞書のようなもの)を生成するプログラム(の超低機能版)を実装してみたのですが、テキストデータを与えても、期待していたような単語ベクトル群(例えば”king” - “queen” = “prince” - “princess”のような関係のあるもの)は得られませんでした。

今回は、極度に問題を単純化、縮小することで、それなりの結果を得ることができたので報告します。



- ゼロから作るDeep Learning ❷——自然言語処理編

OLYMPUS

INTERNAL USE ONLY

01. word2vecとは
02. 失敗原因の考察その1
03. リベンジその1
04. 失敗原因の考察その2
05. リベンジその2
06. まとめ

01 word2vecとは

word2vecとは

指定されたテキストデータをDeepLearningで学習し、テキストデータに含まれる各単語に対し、指定された次元数のベクトル値を設定するもの。

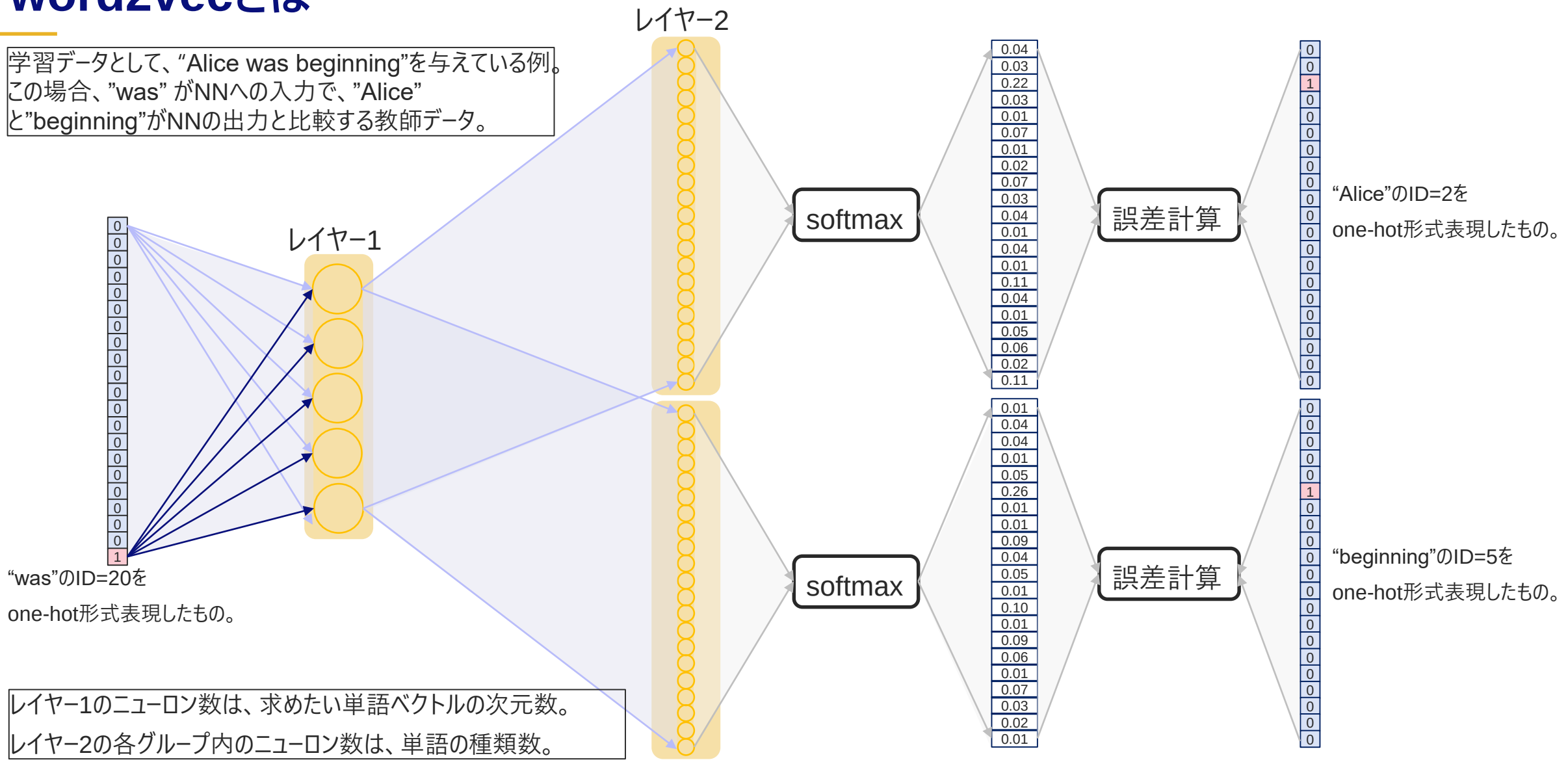
“Alice was beginning to get very tired of sitting by her sister on the bank, and of having nothing to do:”…

単語	1	2	3	4	5
“female”	2.6	7.2	9.4	-5.1	3.7
:					
“king”	1.2	5.5	1.5	-8.2	1.5
:					
“male”	1.9	5.1	1.0	-7.5	5.1
:					
“queen”	1.8	7.9	9.4	-6.2	-0.1
:					

“king” – “male” + “female” = (1.9, 7.6, 9.9 -5.8, 0.1)

word2vecとは

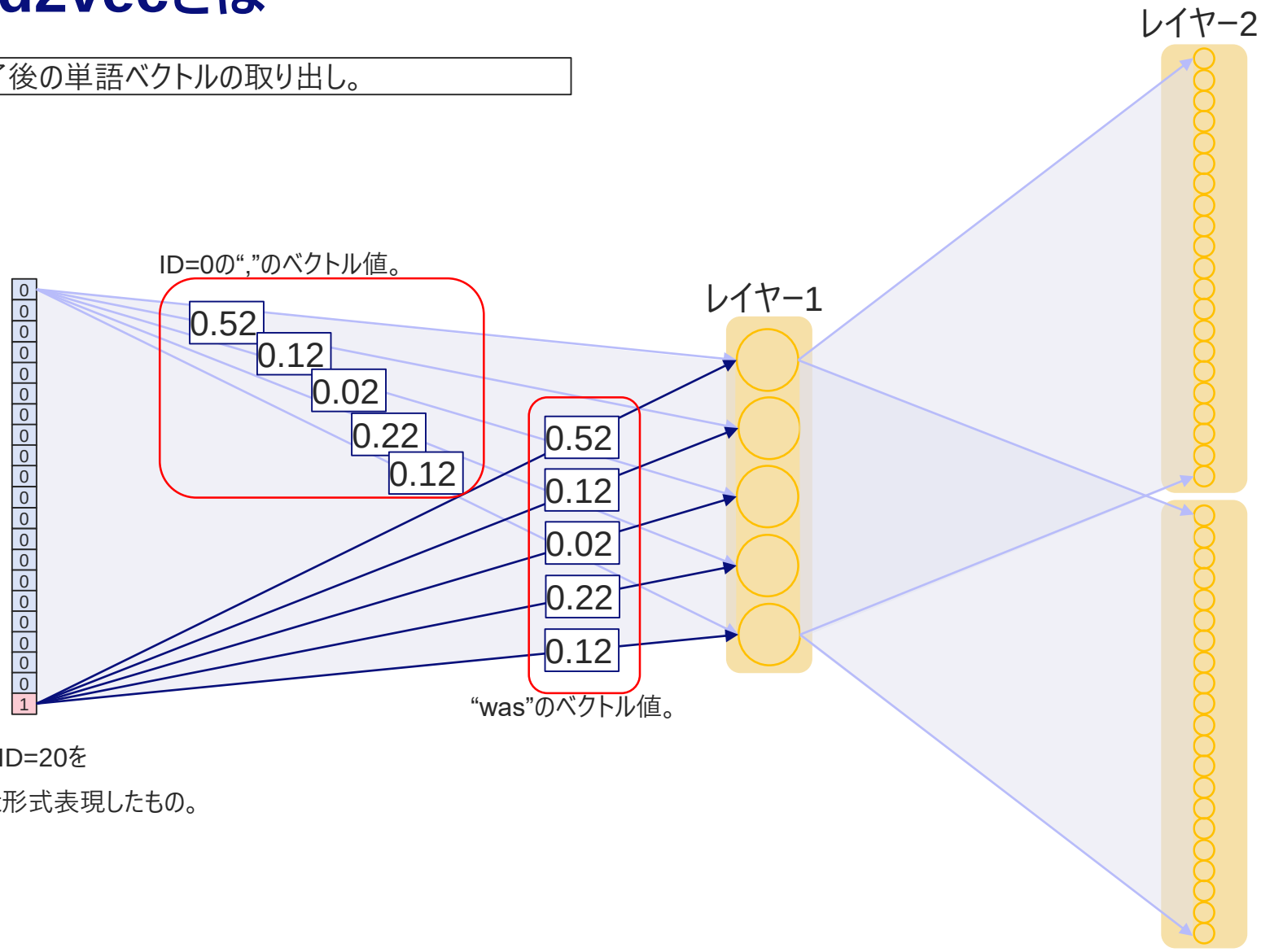
学習データとして、“Alice was beginning”を与えている例。
この場合、”was” がNNへの入力で、”Alice”
と”beginning”がNNの出力と比較する教師データ。



レイヤー1のニューロン数は、求めたい単語ベクトルの次元数。
レイヤー2の各グループ内のニューロン数は、単語の種類数。

word2vecとは

学習終了後の単語ベクトルの取り出し。



“was”のID=20を

one-hot形式表現したもの。

02 失敗原因の考察その1

失敗原因の考察その1

規模がとにかく小さすぎるのではないか?

1. 本家word2vecに比べると、単語ベクトルの次元数が小さい。

   本家word2vecはデフォルトが200であるのに対し、前回の独自実装版は20であった。

2. 本家word2vecに比べると、ウィンドウサイズが小さい。

   本家word2vecはデフォルトが5であるのに対し、前回の独自実装版は1であった。

3. 学習に使用したテキストデータが小さい。

   一般的に非常に大きなもの(Wikipediaとか)を使うらしいが、前回使用したのは、”ALICE’S ADVENTURES IN WONDERLAND”という

   170kbのテキストファイルだった。

03 リベンジその1(途中で断念)

リベンジその1(途中で断念)

本家word2vecに匹敵するものを作成するのはおそらく非常に難しく、あまり意味もないと思われたので、以下のページで公開されている単語ベクトルデータを入手し、「適切に作成された単語ベクトルは実際のところどのようなものなのか?」を調べることにしました。

https://github.com/Kyubyong/wordvectors

このデータは、単語ベクトルが300次元となっており、Excelですらまともに扱えないものであったため、単語の計算式を入力すると、計算結果の単語ベクトルと、これに近い値を持つ単語を上位10個表示するような簡単なプログラムを作成することにしました。

ある程度できたところで、"おはよう - 朝 - 晩"という計算式を入力してみたのですが、以下のような結果しか得られませんでした。("こんばんは"あたりが出力されることを期待していました。)

単語	距離
ではこれらの	1209
の	1388
が	1525
は	1553
と	1615
に	1708
を	1750

04 失敗原因の考察その2

失敗原因の考察その2

自然言語はやはり難しいのではないか?

1. 単語の種類が多い。
2. 単語によっては学習用テキストでの出現頻度が極端に低いものがあり得る。
3. 字面は同じでも、意味の異なる単語が多数ある。
4. 日本語の場合、単語の切れ目が明確でない。
5. 単語ベクトルの性能指標が明確ではない。

05 リベンジその2

リベンジその2

自然言語特有の難しさを排除した、以下のような人工言語を作り、これをword2vecがどのように処理するのかを調べることにしました。

1. 単語は、-9から+9までの整数とする。
2. 学習用データは、二つの数の足し算とその答えとする。
3. 単語ベクトルの次元数は2次元とする。(これは単語ベクトルの可視化を簡単にするため。)

リベンジその2

学習用データ

-9 = -9 + 0
-9 = -8 + (-1)
-9 = -7 + (-2)

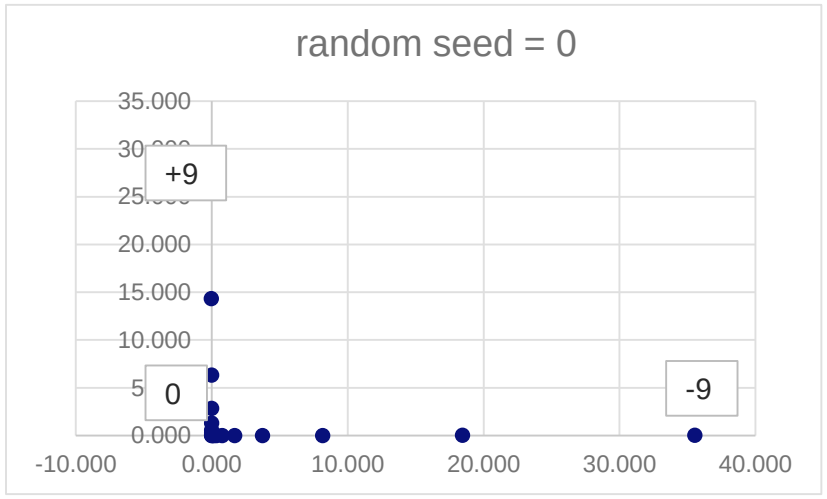
No.	Target	Context1	Context2
1	-9	-9	0
2	-9	-8	-1
3	-9	-7	-2
		:	

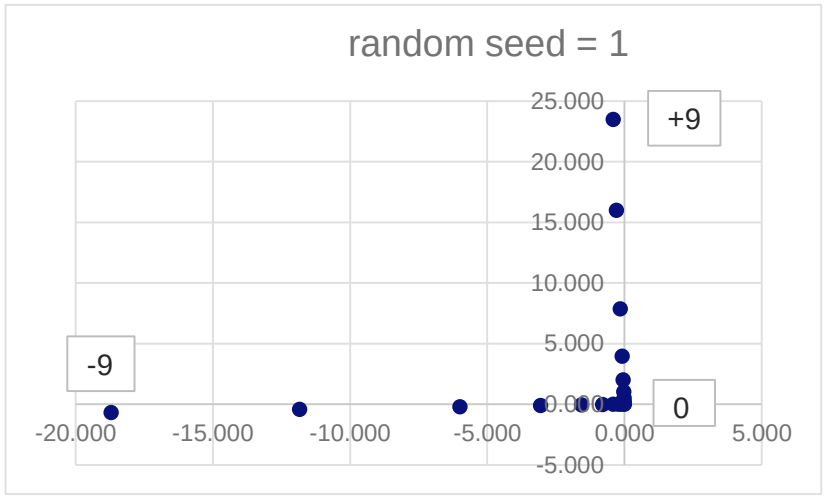
自然言語の場合の学習用データの例

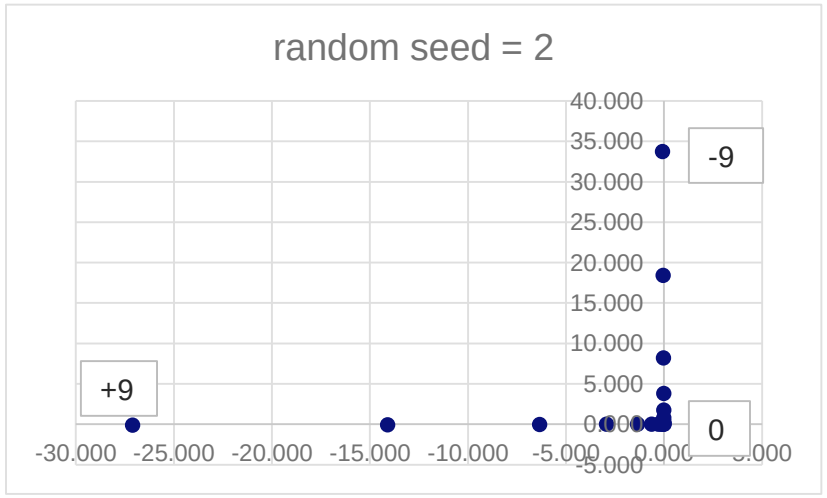
“Alice was beginning to get very tired of sitting by her sister on the bank, and of having nothing to do:”…

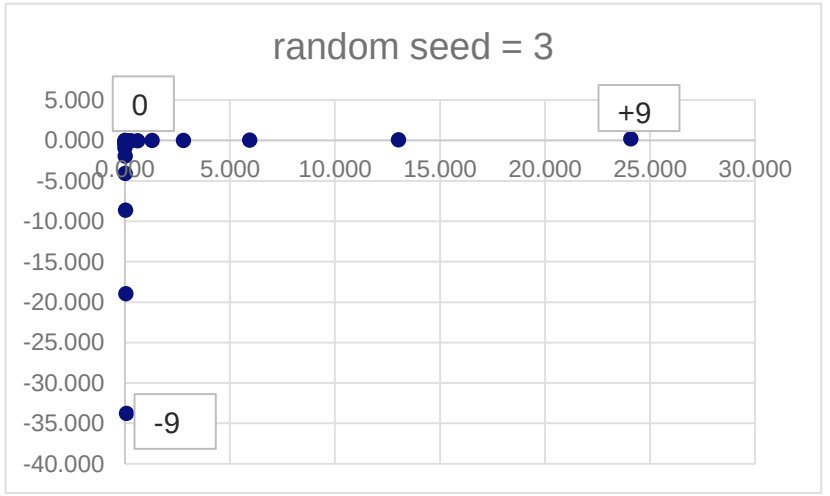
No.	Target	Context1	Context2
1	was	Alice	beginning
2	beginning	was	to
3	to	beginning	get
		:	

リベンジその2



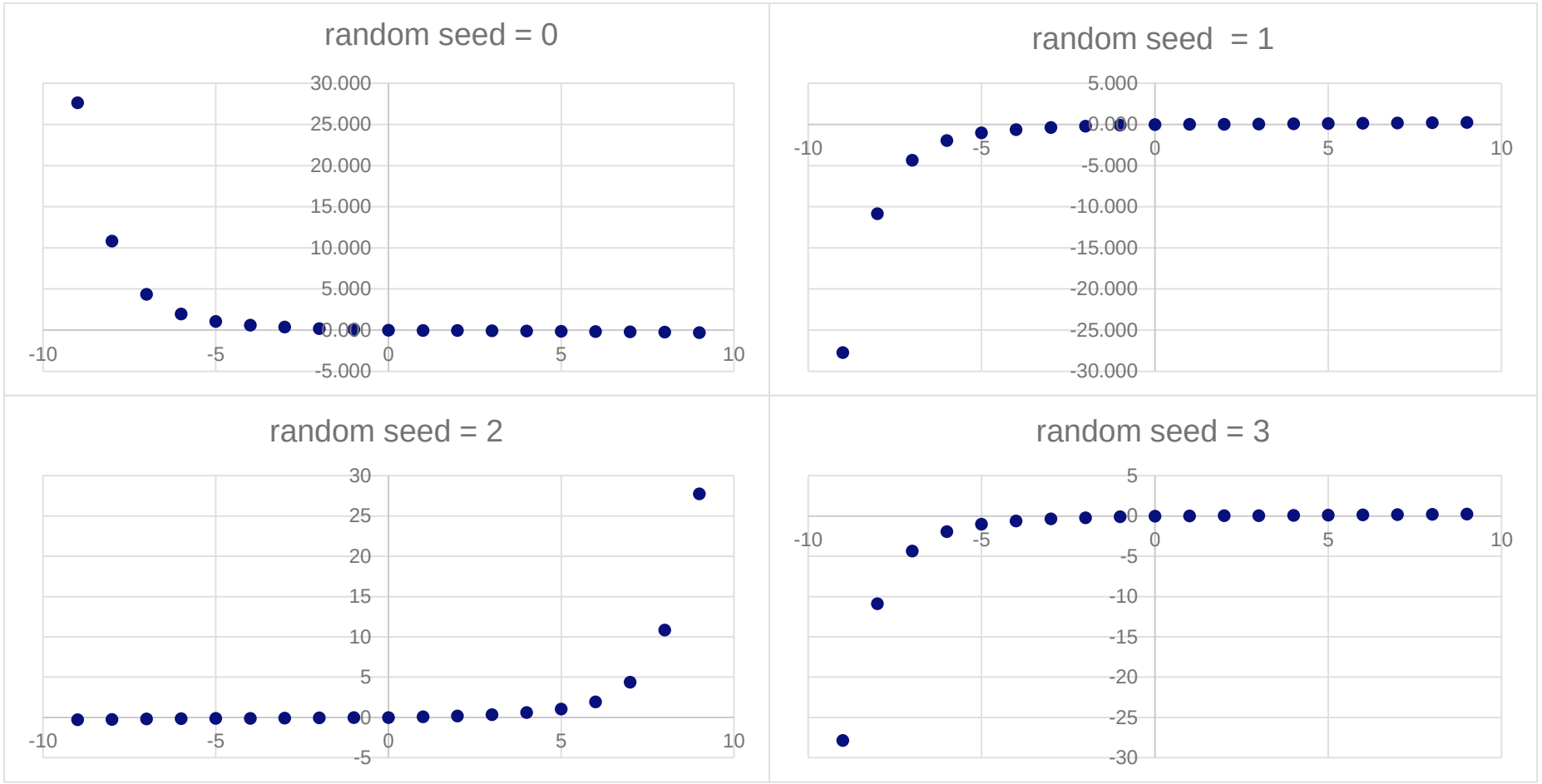






リベンジその2

単語ベクトルのサイズを2次元から1次元に変更



OLYMPUS

06 まとめ

まとめ

「-1の単語ベクトル + -2の単語ベクトル」が、「1の単語ベクトル + -4の単語ベクトル」のような関係こそ見られませんでしたが、単語ベクトルを2次元とした場合は、0の単語ベクトルはほぼ(0, 0)になり、正の数と負の数がそれぞれ異なる軸にのり、順序も維持されるらしいという結果が得られ、同様に1次元とした場合でも、0の単語ベクトルが(0)になり、値の順序関係も維持されるらしいという結果も得られました。

今回の発表では、なぜこのような結果が得られるのかは説明できていないので、もう少し追究していきたいと思います。


OLYMPUS