推論攻撃の情報漏洩に着目した言語ベクトルの次元圧縮

木下研究室 201604458 森 大樹

研究背景

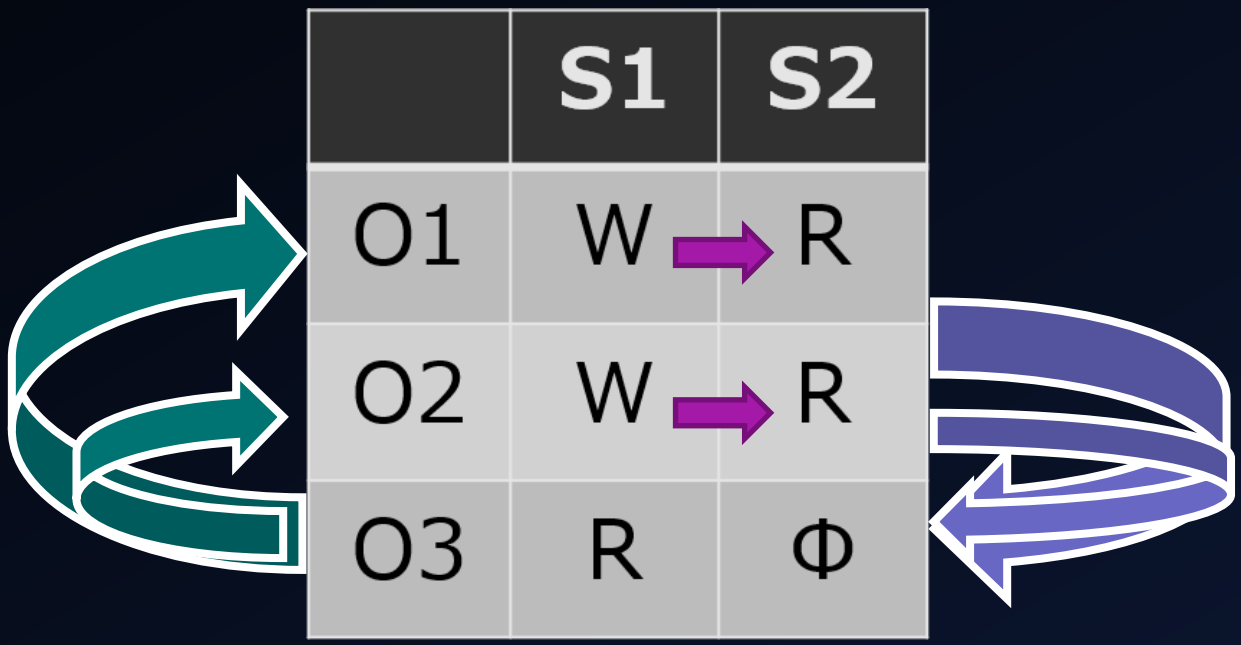
情報セキュリティの中でも対推論攻撃に対して研究がある.

手法としてテクストにおける機械学習を用いた方法がある.

- 単語を独立事象として扱い, 単語が含まれるテクストとの関連において次元を圧縮しているLDA
- ベクトル表現を用いるものとして高次元ベクトルの内積（類似度）を計算するword2vecがある.

推論攻撃とは

O1、O2にO3に類似する内容であった場合、O3にアクセス権がなくともS2はO3の内容を推論できてしまう.



問題点

推論攻撃に対策として自然言語処理を用いるには、そこに推移率が成り立っていなければならない.つまり、
言語の相関を扱えなければならない.

- LDA等のトピック分類はデータは低次元であるが相関がない.
- 相関関係を単語同士の類似を用いてを扱うことのできるSkip-gramやC-BOWではデータが高次元になってしまう.

データの高次元化は速度面、精度面において無視できない問題となっている.

目的

- 本研究の目的としては、日本語ドキュメントにおける自然言語処理による高次元ベクトルデータの次元圧縮である.

研究の方向性

- 今回の次元圧縮では速度ではなく今回は精度に主眼を置く.

提案手法

本研究では、次元圧縮の手法にTucker分解を組み合わせることで精度の向上を目指したものである.手法として

a. **Tucker分解+主成分分析(PCA)**
b. **Tucker分解+t分布型確率的近傍埋め込み(t-SNE)**

これら2種の次元圧縮の手法を用いて、3次元に次元圧縮した後に、圧縮元となった高次元データと比較する.

比較条件

1. **類似単語の一致率**
2. **類似順位が守られているかどうか**

この2点に着目し、精度の比較とした.

高次元ベクトルデータの用意

1. Wikipediaより日本語ウィキペディアのダンプファイルを入手.
2. MeCabを用いて、分かち書き形式に直す.この際に、**名詞のみの抽出**を行う.
3. Word2vecを用いてSkip-gramの100次元の単語ベクトルデータを生成する.

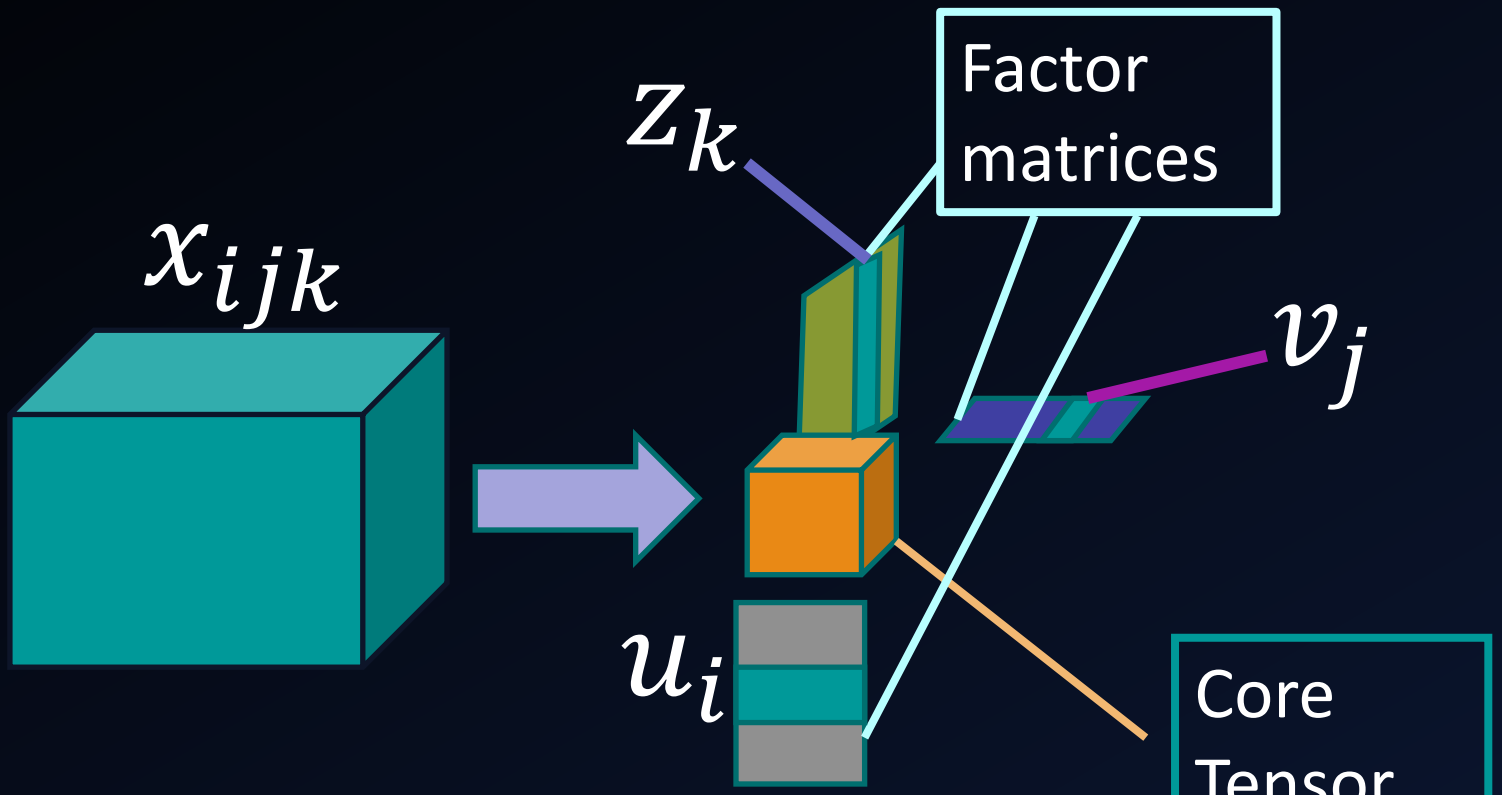
次元圧縮の手順

1. Word2vecを用いて高次元の単語ベクトルを生成する.
2. Numpy > 高次元ベクトルデータをTucker分解
3. Tensorflow > 主成分分析、t分布型確率的近傍埋め込みを用いて3次元化
4. Tensorboard > Tensorの表示

Tensor分解(Tucker分解)

Tensor分解とは、Tensorを二つに分解して考えるというものである.高次元のTensorであっても、行列で表すことが可能となる.Tucker分解もTensor分解の手法の
一つであり、**TensorをコアTensorと基底となる行列の積に分解する.**

Tensor分解(Tucker分解)



$$x_{ijk} = \sum_{q} t_{qrs} u_{iq} v_{jr} z_{ks} + \varepsilon_{ijk}$$

提案手法(Tucker分解+PCA)

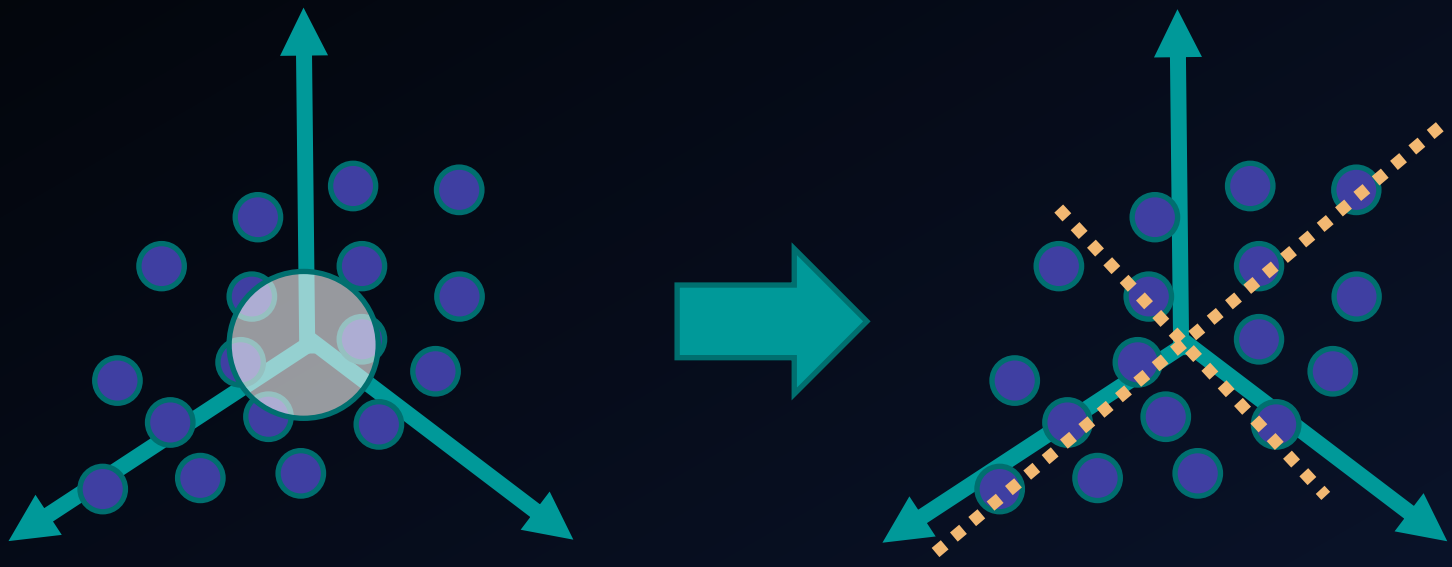
1. Word2vecを用いて高次元の単語ベクトルを生成する.

2. 高次元ベクトルをTucker分解を用いて、コアテンソルと基底となる行列の積に分解する.
3. ベクトルデータの重心(平均)を求める.
4. 分解されたTensorから分散の高い(主成分)軸を求める.
5. 求めた主成分軸に直行する軸で新たな軸を求める.

以上を繰り返し行うことで次元の圧縮を行う.

手順、2～4がPCAの手順

提案手法(Tucker分解+主成分分析(PCA))



データの平均(重心)

主成分軸を求める

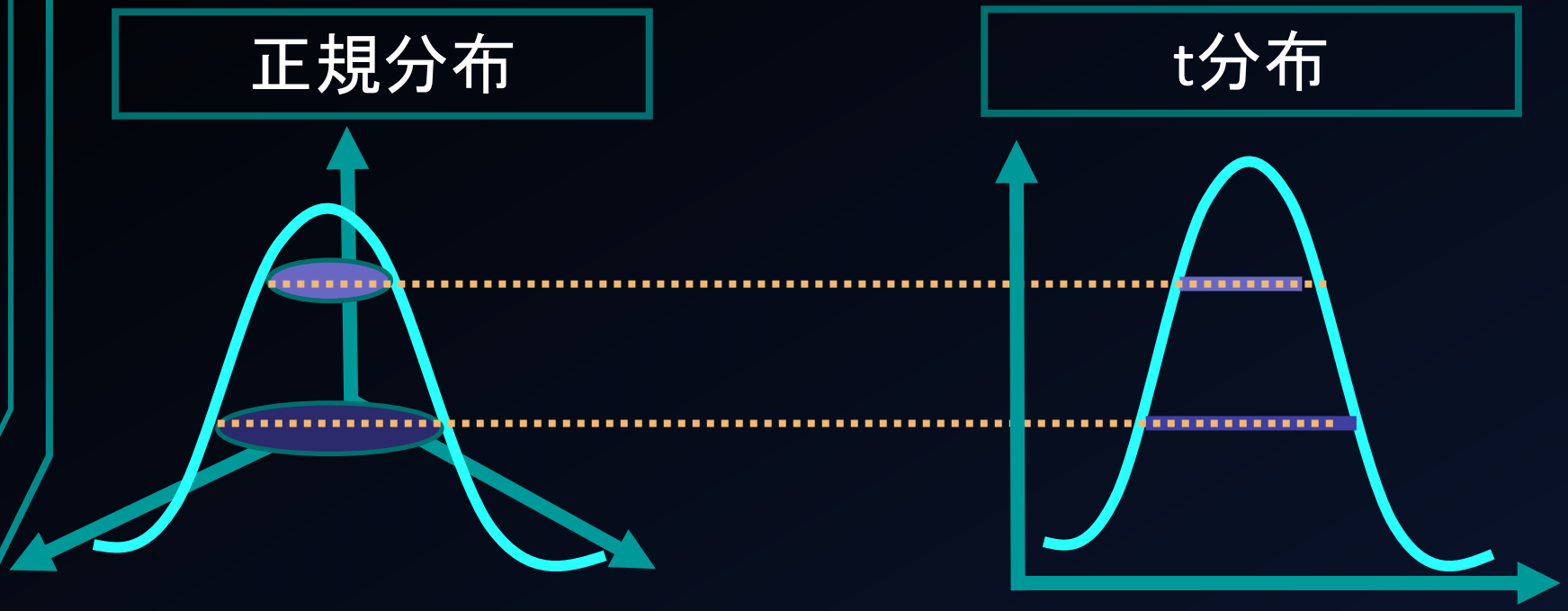
提案手法(Tucker分解+t-SNE)

1. Word2vecを用いて高次元の単語ベクトルを生成する.

2. 高次元ベクトルをTucker分解を用いてコアテンソルと基底となる行列の積に分解する.
3. 正規分布に従って高次元の点の間の距離を計算する.
4. 高次元の各点のPerplexity が任意のレベルになるように、各点 i について標準偏差 σi を作成する.
5. t 分布に従って、低次元の点の初期集合を作成する.
6. 高次元空間の正規分布と低次元空間のt分布のKL情報量が最小になるように、低次元の点を繰り返し更新する.

手順、2～5がt-SNEの手順

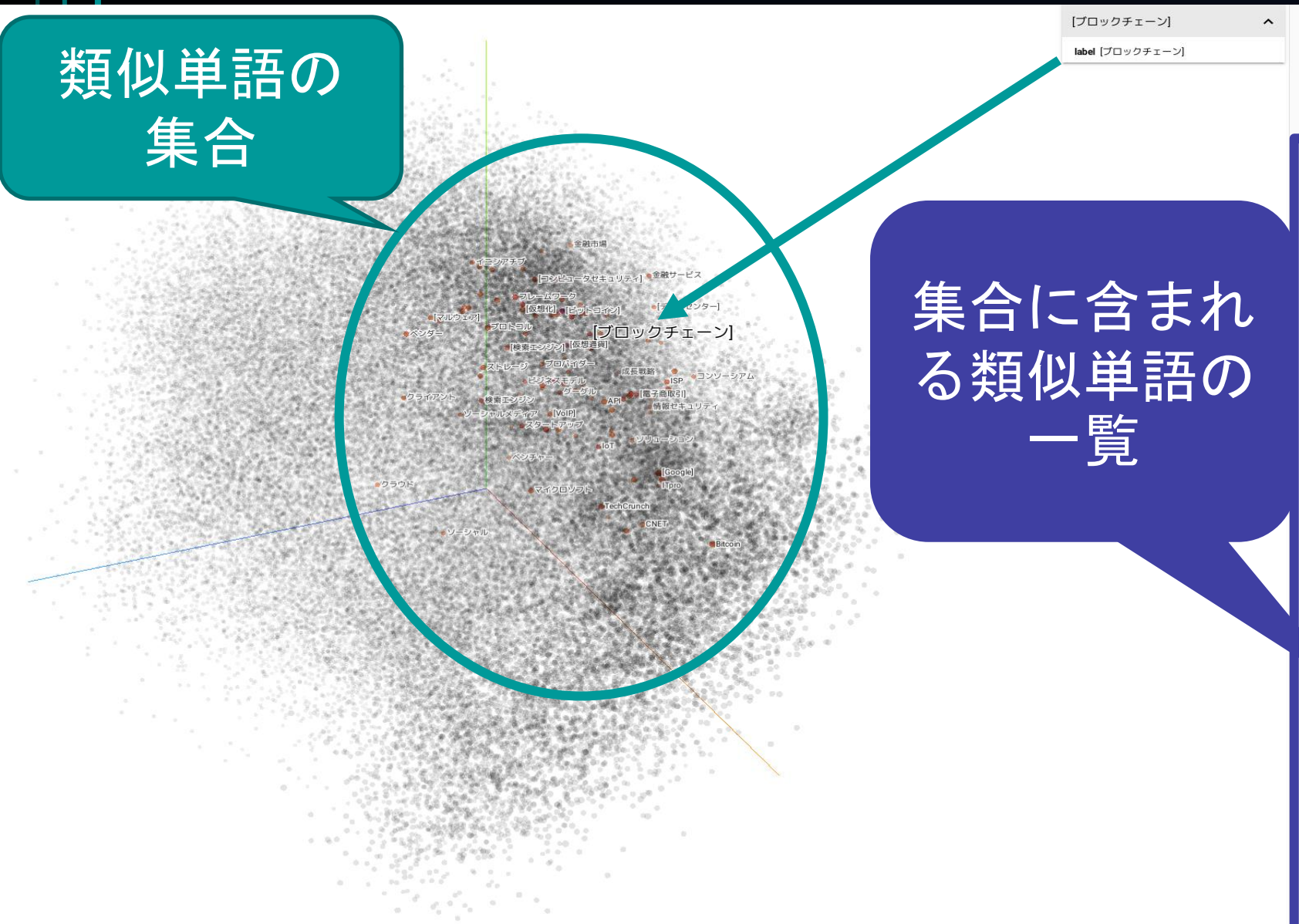
提案手法(Tucker分解+ t分布型確率的近傍埋め込み)



KL情報量が小さくなるように各点を計算

次元圧縮したモデルのTensorboardによる出力

類似単語の集合



集合に含まれる類似単語の一覧

実験結果

	word	類似度 経過時間
2	('[ブロックチェーン]'	0.9083224534988403)
3	('暗号通貨'	0.8612357974052429)
4	('[フィンテック]'	0.8216238617897034)
5	('スマートコントラクト'	0.8191790580749512)
6	('[SAP_BW/4HANA]'	0.8171334266662598)
7	('[仮想通貨]'	0.8136426210403442)
8	('[ロボティック・プロセス・オートメーション	0.8113190531730652)
9	('FinTech'	0.8101117014884949)
10	('[情報銀行]'	0.8094927072525024)
11	('アナリティクス'	0.8094843029975891)
12	The required time	115.991 sec.
13	('[ネットワーク]'	0.8043789863586426)
14	('[コンピュータネットワーク]'	0.7524142265319824)
15	('双方向'	0.7378183603286743)
16	('[インターネットバックボーン]'	0.7338780164718628)
17	('[UUCP]'	0.732590973772278)
18	('[オーバーレイ・ネットワーク]'	0.7311298847198486)
19	('[広域イーサネット]'	0.7303845882415771)
20	('[エクストラネット]'	0.7255366444587708)
21	('[Wide_Area_Network]'	0.7250081300735474)
22	('ネットワークインフラ'	0.7228484749794006)
23	The required time	116.013 sec.
24	('納豆'	0.8771916031837463)
25	('梅干'	0.8246580958366394)
26	('食べ物'	0.8174327611923218)
27	('ピーマン'	0.8169663548469543)
28	('お菓子'	0.8111935257911682)
29	('焼きうどん'	0.8107301592826843)
30	('豆腐'	0.8094905018806458)
31	('肉じゃが'	0.8079715967178345)
32	('りんごジュース'	0.804995596408844)
33	('高野豆腐'	0.8033233284950256)
34	The required time	116.032 sec.
35	('[シリコン]'	0.8980396389961243)
36	('アモルファスシリコン'	0.8638802170753479)
37	('絶縁体'	0.8519055843353271)
38	('薄膜'	0.8477480411529541)
39	('磁性体'	0.8384630680084229)
40	('ポリイミド'	0.8381196856498718)
41	('[薄膜]'	0.8380274772644043)
42	('電極'	0.8363238573074341)
43	('半導体'	0.8359363079071045)
44	('ウエハー'	0.8352245688438416)
45	The required time	116.058 sec.
46	('うどん'	0.894124448299408)
47	('焼きそば'	0.8923447728157043)
48	('麺'	0.8876796960830688)
49	('[ラーメン]'	0.8866945505142212)
50	('チャーハン'	0.8766574859619141)
51	('餃子'	0.8741289973258972)
52	('らーめん'	0.87374347448349)
53	('つけ麺'	0.8679471611976624)
54	('丼'	0.8662068843841553)
55	('味噌ラーメン'	0.8645495176315308)
56	The required time	116.085 sec.
57	('神奈川県'	0.8186288475990295)

単語の一致率

100次元時の類似単語上位10個と3次元時の類似単語上位20個の類似単語の一致率.

類似度順位の維持率

一致した単語における、100次元時と3次元時の類似度の順序の一致率.

実験結果

	Tucker分解 + PCA	Tucker分解 + t-SNE
類似単語の 一致率	16.7%	25.3%
類似度順位の維持 率	86.9%	92.2%

まとめ

日本語の単語は使用される場面によって異なる単語が選択されるため、

- Tucker分解により意味的まとまりを取り出す.
- 更に主成分軸を用いた圧縮法であるPCA .
- 局所構造の維持, 意味的に遠い単語を遠くに配置することのできるt-SNE.

これらの手法は意味的な高次元まとまりを低次元に再現することできるのであろう.

推論攻撃における本実験のまとめ

高次元の単語ベクトルデータを次元圧縮することで相関関係を満たす低次元かつ高精度の単語ベクトルデータの算出することができた. これにより高次元ゆえに問題になっていた重みづけの難しさなどが解消され, 今後情報セキュリティーにおける推論規則の生成に有効である.