N-gram

検索の潮流

形態素解析不要！

N-gram

- 文字列をN文字単位の文字列片に分解する
- 文字列片を見出しとしたインデックスを作成する
- 文字列片の出現頻度を計算する
- 出現頻度の類似性を評価する

N-gram, N=2

- 文字列を２文字単位の文字列片に分解する
- 文字列片を見出しとしたインデックスを作成する
- 文字列片の出現頻度を計算する
- 出現頻度の類似性を評価する

N-gram, N=2

「母は歯が悪い」

インデックス	頻度
母は	1
は歯	1
歯が	1
が悪	1
悪い	1

「母は歯が悪い」「母は間が悪い」

インデックス	頻度
母は	1
は歯	1
歯が	1
が悪	1
悪い	1

インデックス	頻度
母は	1
は間	1
間が	1
が悪	1
悪い	1

「母は歯が悪い」「母は間が悪い」

インデックス	頻度
母は	1
は歯	1
は間	0
歯が	1
間が	0
が悪	1
悪い	1

インデックス	頻度
母は	1
は歯	0
は間	1
歯が	0
間が	1
が悪	1
悪い	1

「母は歯が悪い」

インデックス	頻度
母は	1
は歯	1
は間	0
歯が	1
間が	0
が悪	1
悪い	1

A

「母は間が悪い」

インデックス	頻度
母は	1
は歯	0
は間	1
歯が	0
間が	1
が悪	1
悪い	1

B

```python
import numpy as np

A = np.array([1, 1, 0, 1, 0, 1, 1])
B = np.array([1, 0, 1, 0, 1, 1, 1])
```

```python
# L1 norm : 各次元の値の絶対値の和「マンハッタン距離」
print(np.linalg.norm(A-B, ord=1))

# L2 Norm : 各次元の値を2乗した和の平方根「ユークリッド距離」
print(np.linalg.norm(A-B, ord=2))
```

```
4.0
2.0
```

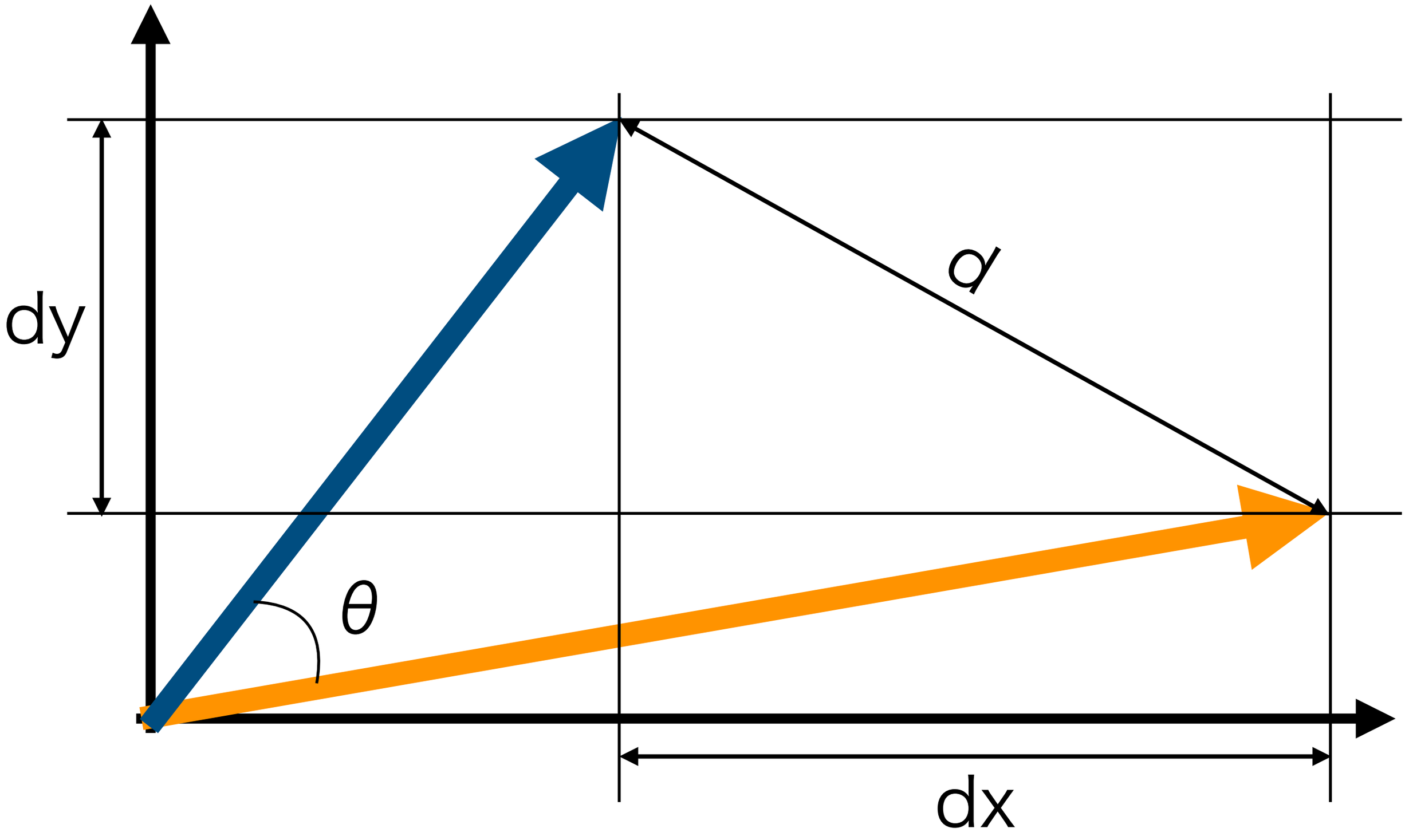
```python
# Cos類似度：２つのベクトルのなす角度のCos：
# 一致していたら１。９０度違ったら０．１８０度違ったら－１
def cos_sim(v1, v2):
    return np.dot(v1, v2) / (np.linalg.norm(v1) * np.linalg.norm(v2))
```

```python
print(cos_sim(A, B))
```

```
0.5999999999999999
```

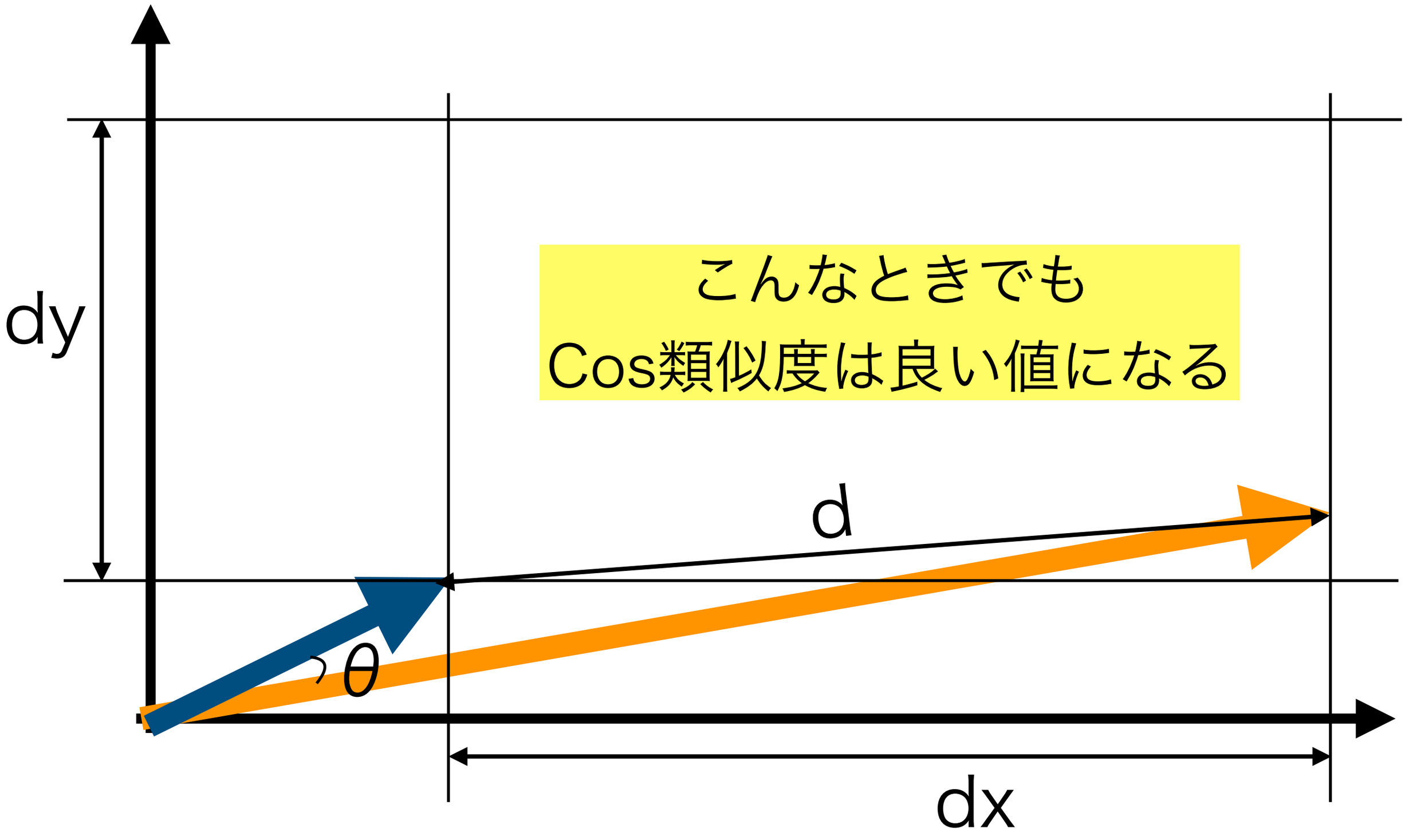
L1, L2, Cos類似度

L1 norm: dx + dy, L2 norm: d, Cos類似度：$\cos\theta$



L1, L2, Cos類似度

L1 norm: dx + dy, L2 norm: d, **Cos類似度：cos θ**



word2vec

特徴抽出の潮流

形態素解析不要だけど
モデル作成の時には形態素解析済みの
ファイルを利用するよ

word

歯　が　悪い
間　が　悪い
目　が　悪い

入力

vec

単語の連続性を学習
（前後の単語を予測）

特徴

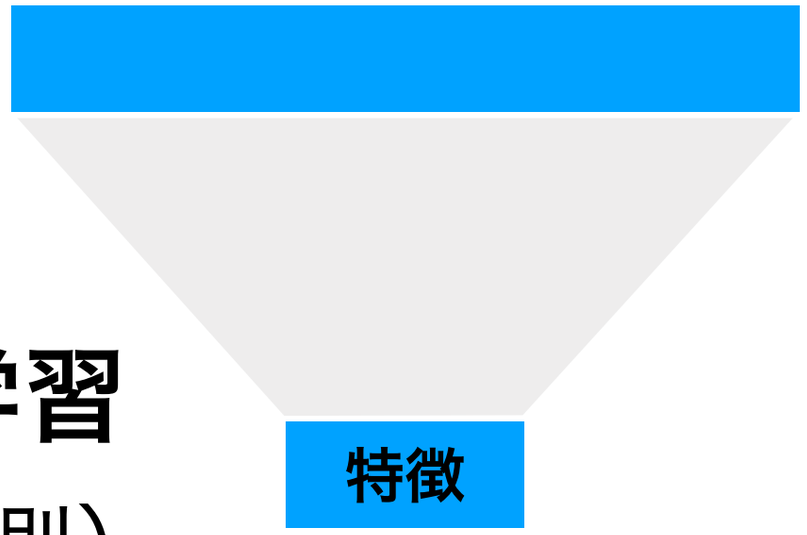
出力　出力　出力

歯　が　悪い　　間　が　悪い　　目　が　悪い

word

word

歯　が　悪い
間　が　悪い
目　が　悪い



vec

単語の連続性を学習
（前後の単語を予測）

word2vec

```python
result1 = model.wv["書籍"]
print(result1)
print(result1.shape)
```

```
[ 6.2136418e-01  1.2860003e+00 -9.6988064e-01  3.1051097e+00
 -1.5730866e+00  6.3269925e-01 -9.3587315e-01  2.2156084e+00
 -1.9908961e+00  4.7987399e+00 -2.1442327e+00  3.3477969e+00
  8.3119637e-01 -3.9228864e+00 -9.2465019e-01 -5.9408846e+00
  3.2142821e-01 -1.9636788e+00 -4.6879988e+00 -7.6191908e-01
  3.5126407e+00  2.1188366e+00 -5.8922863e-01 -2.2432396e+00
  3.2605505e-01  4.3294644e+00  7.7272615e+00  1.4935368e-01
 -2.0937143e-01 -2.8820115e-01  1.2384745e+00 -5.6035876e+00
 -3.7708589e-01 -3.5765567e+00 -5.8662143e+00  6.0884045e-03
 -3.0726111e+00  4.7189765e+00 -1.1070979e+00  5.3821278e-01
 -1.2472039e+00  5.6540662e-01 -1.3219589e+00 -4.9341077e-01
 -8.7207955e-01 -5.3856981e-01 -2.6360452e+00  5.1414204e-01
  3.4013101e-01 -7.7331238e+00 -3.7250347e+00  3.1997075e+00
 -5.6322141e+00 -5.9097214e+00  8.3416390e-01  4.8727946e+00
  7.4329299e-01 -7.2315282e-01 -4.7385564e+00 -1.8719325e+00
  5.8828893e+00  2.4761300e-01 -3.3648055e+00 -2.2935954e-01
  2.8681259e+00  1.7134066e+00  1.9100230e+00 -1.7395239e+00
  1.7555764e-01 -2.5397058e+00 -3.5353439e+00 -1.4471033e+00
 -6.6979721e-02 -5.0519834e+00  8.3071750e-01  9.3190843e-01
  5.6917912e-01 -4.3857780e-01 -5.8661556e-01 -4.3340549e+00
 -2.8615551e+00 -1.5810206e+00  8.2434362e-01  1.7237933e+00
  2.0397053e+00 -1.2942677e+00 -4.7680148e-01  5.7332903e-01
 -3.6395478e+00  1.8429600e+00 -1.8048346e+00 -1.1689606e+00
 -1.2700789e+00 -8.4880781e+00 -1.5261092e+00 -4.3575859e+00
  5.1227921e-01  5.7538886e+00 -8.1334138e-01 -1.4806017e+00]
(100,)
```

```
result1 = model.wv["書籍"]
print(result1)
print(result1.shape)
```

```
[ 6.2136418e-01  1.2860003e+00 -9.698
 -1.5730866e+00  6.3269925e-01 -9.358
 -1.9908961e+00  4.7987399e+00 -2.144
  8.3119637e-01 -3.9228864e+00 -9.246
  3.2142821e-01 -1.9636788e+00 -4.687
  3.5126407e+00  2.1188366e+00 -5.892
  3.2605505e-01  4.3294644e+00  7.727
 -2.0937143e-01 -2.8820115e-01  1.238
 -3.7708589e-01 -3.5765567e+00 -5.866
 -3.0726111e+00  4.7189765e+00 -1.107
 -1.2472039e+00  5.6540662e-01 -1.321
 -8.7207955e-01 -5.3856981e-01 -2.638
  3.4013101e-01 -7.7331238e+00 -3.725
 -5.6322141e+00 -5.9097214e+00  8.341
  7.4329299e-01 -7.2315282e-01 -4.738
  5.8828893e+00  2.4761300e-01 -3.364
  2.8681259e+00  1.7134066e+00  1.910
  1.7555764e-01 -2.5397058e+00 -3.535
 -6.6979721e-02 -5.0519834e+00  8.307
  5.6917912e-01 -4.3857780e-01 -5.866
 -2.8615551e+00 -1.5810206e+00  8.243
  2.0397053e+00 -1.2942677e+00 -4.768
 -3.6395478e+00  1.8429600e+00 -1.804
 -1.2700789e+00 -8.4880781e+00 -1.526
  5.1227921e-01  5.7538886e+00 -8.133
(100,)
```

```
result2 = model.wv["図書"]
print(result2)
print(result2.shape)
```

```
[  7.3487196    3.428948     0.3767956    4.474103
   2.883772    -0.398988    -0.4069058   -0.9142845
  -4.4536595    1.8984295   -2.028987     1.6110324
  -4.6914687    2.9567707   -3.5218427   -2.918899
  -0.16187252  -0.11621725   0.10565718  -4.3653555
   1.048472     2.9977999    0.04923692  -3.4863486
  -3.8627553   -1.2002124    0.932865     1.6199659
  -3.2934268   -3.2426286    4.400625    -2.3534834
  -2.397414     3.5938163   -3.1743646   -1.575356
  -1.0887389   -4.883394     5.9480586   -0.36048728
  -1.5157106    0.67435014  -6.3831673  -11.472011
   8.427529     5.1425414   -3.3770251   -2.2621117
   3.673554     3.253771     2.5279279   -5.75144
  -0.71290785  -0.80596775  -0.6379076    0.5821562
  -0.2639271   -1.7041026    0.4985876   -4.3205004
   4.2915277   -0.4942163    1.743835     2.1020634
   2.1519358   -0.6606972    0.66042715  -2.3667285
  -0.2518153    0.29362133   0.75548524  -5.883023
  -5.244054     1.4057037    0.56451017  -1.0581326
   0.93234986  -1.2591007    2.4999673    0.6302473
(100,)
```

```python
import numpy as np

# L1 norm : 各次元の値の絶対値の和 「マンハッタン距離」
L1norm = np.linalg.norm(result1-result2, ord=1)

# L2 Norm : 各次元の値を2乗した和の平方根 「ユークリッド距離」
L2norm = np.linalg.norm(result1-result2, ord=2)

print(L1norm)
print(L2norm)
```

```
240.29868
29.571907
```

```python
# Cos類似度：２つのベクトルのなす角度のCos
# 一致していたら１。９０度違ったら０．１８０度違ったら－１
def cos_sim(v1, v2):
    return np.dot(v1, v2) / (np.linalg.norm(v1) * np.linalg.norm(v2))
```

```python
print(cos_sim(result1, result2))
```

```
0.5255668
```