Word2Vecを用いた文章問題採点システムの試作

日本大学文理学部　情報科学科　谷聖一研究室　坂本 祐一郎

2019/1/31

目次

はじめに

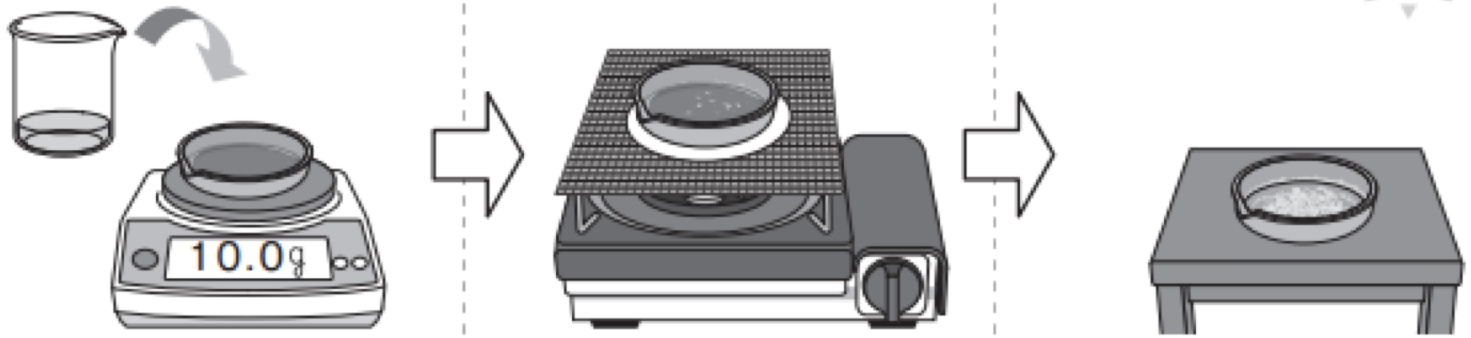
小学校理科の文章問題において，
答えが複数存在するものがある

問題(例)国立教育政策研究所
平成30年度全国学力・学習状況調査 小学校6年生 理科

【問題】 食塩水の食塩は，蒸発（じょうはつ）するのだろうか。

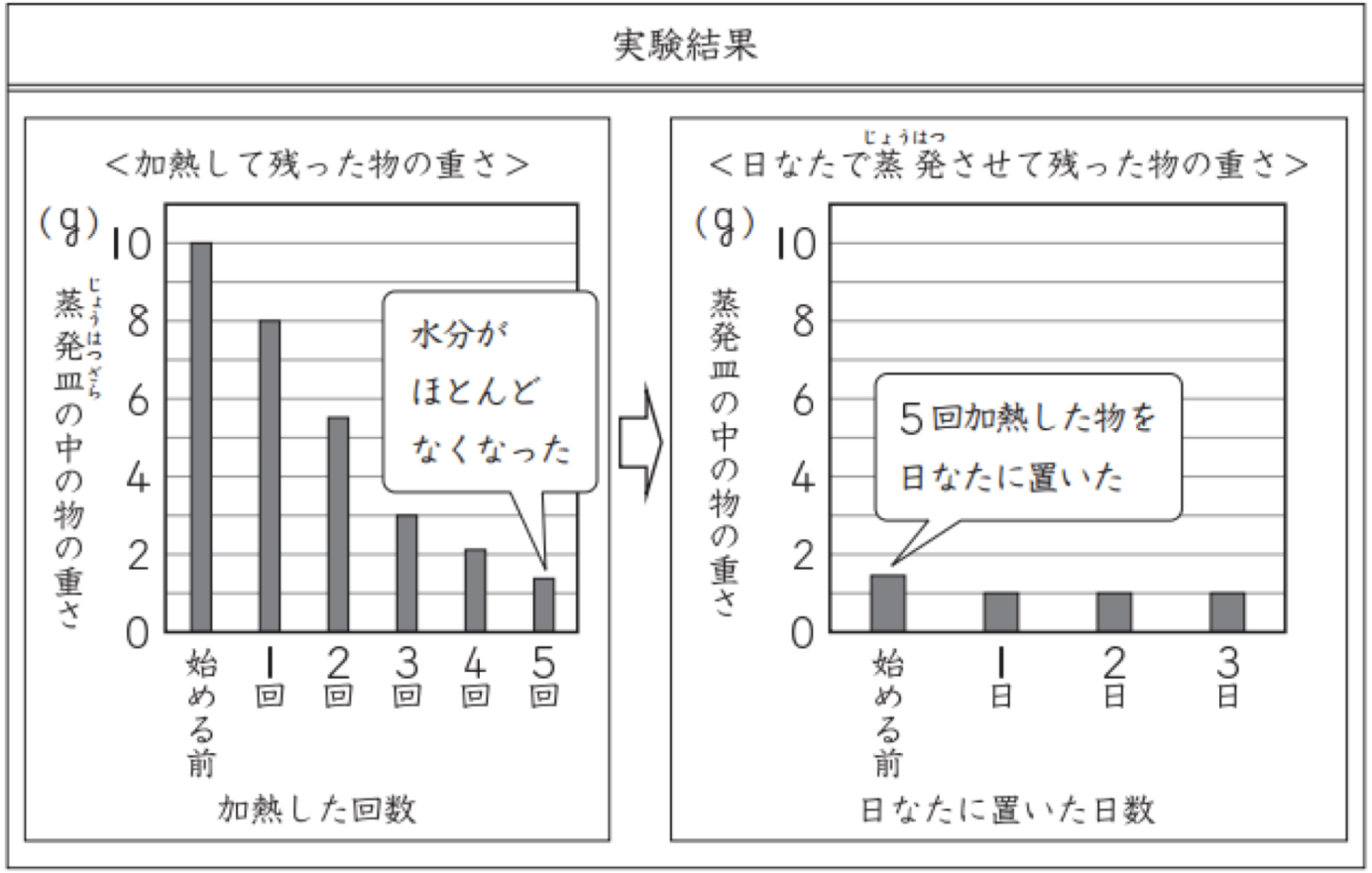
実験方法

1gの食塩に水を加えて10gにした食塩水すべてを蒸発皿（じょうはつざら）に入れる。	実験用ガスコンロで1分間加熱し，冷ましてから重さをはかる。水分がほとんどなくなるまで，くり返す。	日なたに置いて蒸発（じょうはつ）させ，1日ごとに重さをはかる。



(引用元：
国立教育政策研究所
平成30年度全国学力・学習状況調査，
http://www.nier.go.jp/18chousa/18chousa.htm)

問題(例)
国立教育政策研究所 平成30年度全国学力・学習状況調査 小学校6年生 理科



ゆかりさんは，実験の結果からいえることを，下のようにまとめました。

【実験の結果からいえること】 水にとけた物は蒸発しない。

(引用元：
国立教育政策研究所
平成30年度全国学力・学習状況調査,
http://www.nier.go.jp/18chousa/18chousa.htm)

問題(例)国立教育政策研究所
平成30年度全国学力・学習状況調査 小学校6年生 理科

(4) ゆかりさんが**【実験の結果からいえること】**としてまとめた内容は,

【問題】に対するまとめとしてふさわしくありません。

ふさわしいまとめになるように書き直しましょう。

(引用元：
国立教育政策研究所
平成30年度全国学
力・学習状況調査,
http://www.nier.go.
jp/18chousa/18ch
ousa.htm)

動機

先の問題の模範解答

食塩水の食塩は，蒸発しない

(引用元：国立教育政策研究所 平成30年度全国学力・学習状況調査,
http://www.nier.go.jp/18chousa/18chousa.htm)

動機

模範解答の他にも正答となるものは存在する

・食塩は，蒸発しない

・食塩水の水は蒸発するが，食塩は蒸発しない

・食塩水の水を蒸発させると，食塩は蒸発しない

・食塩水の水を蒸発させると，水は蒸発するが，食塩は蒸発しない

・溶かした食塩はすべて残っているので，食塩は蒸発しない

・溶かした食塩はすべて残っているので，食塩水の水を蒸発させると，食塩は蒸発しない

・食塩水の水は蒸発するが，溶かした食塩はすべて残っているので食塩は蒸発しない

・溶かした食塩はすべて残っているので，食塩水の水を蒸発させると，水は蒸発するが，食塩は蒸発しない

(引用元：国立教育政策研究所 平成30年度全国学力・学習状況調査， http://www.nier.go.jp/18chousa/18chousa.htm)

これらも正解となる

動機

このような文章問題を人力で採点する場合の問題点

- 採点者による正誤判定基準のぶれ
- 量が多いと，時間がかかる

動機

解答が複数考えられる文章問題を，
機械で正解か不正解か判定できるのだろうか？

目的

小学校理科テストの文章問題の解答同士の類似度を算出し，その類似度を用いて正誤判定を行う

演習の流れ

Webページからデータの収集

Word2Vecを用いて単語の分散表現を獲得

獲得した分散表現の平均ベクトルを用いて文章をベクトル化

文章同士の類似度を算出するプログラムの作成

準備

類似度

類似度とは

　2つのものがどれだけ似ているかを表す指標

類似度の例

- cos類似度
  「類似度のひとつで，ベクトルの内積を用いて類似度を計算する方法」
  (引用元:統計Web，https://bellcurve.jp/statistics/glossary/5488.html，2019/1/23)
- ピアソンの積率相関係数
  「2つの量的変数間の直線的関連の程度を表す係数で，
  いわゆる相関係数のことを示す」
  (引用元:統計Web ，https://bellcurve.jp/statistics/glossary/1233.html，2019/1/23)
- Jaccard係数
  「2つの集合に含まれている要素のうち共通要素が占める割合」
  (引用元:MIERUCA，https://mieruca-ai.com/ai/jaccard_dice_simpson/，2019/1/23)

類似度

本演習で用いた類似度

cos類似度

文書同士の類似度を測る際に，よく用いられる

cos類似度

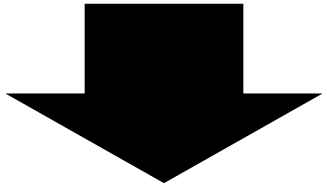
2つのベクトル $\vec{a} = (a_1, a_2 \cdots a_n)$ と $\vec{b} = (b_1, b_2 \cdots b_n)$ に対して

$$cos類似度 = \frac{\vec{a} \cdot \vec{b}}{|\vec{a}| \cdot |\vec{b}|} = \frac{a_1 b_1 + \cdots + a_n b_n}{\sqrt{a_1^2 + \cdots + a_n^2}\sqrt{b_1^2 + \cdots + b_n^2}}$$

値が 1 に近いほど類似度が高く -1 に近いほど類似度が低い

文章のベクトル化

cos類似度では対象が数値ベクトルで表現されていることが前提



文章をベクトル化したい

文章のベクトル化手法

代表例

- Doc2Vec
- tf-idf
- 文章中の各単語の分散表現の平均

本演習で用いたもの

文章中の各単語の分散表現の平均

分散表現

「分散表現（あるいは単語埋め込み）とは，
単語を高次元の実数ベクトルで表現する技術単語の分散表現は
一般に 200次元などの高い次元のベクトルで表されます [1]」

単語をベクトルで表現すると
- 足し算や内積など演算が可能
- 類似度を測れる

[1](引用元:岩波データサイエンス，
https://sites.google.com/site/iwanamidatascience/vol2/word-embedding，2019/1/24)

分散表現(よく用いられる例)

“王様” − “男” ＋ “女” ＝ ？ [1]

[1] Tomas Mikolov , Kai Chen , Greg Corrado , Jeffrey Dean
Efficient Estimation of Word Representations in Vector Space
https://arxiv.org/pdf/1301.3781

分散表現(よく用いられる例)

仮に、4次元のベクトルで表現したとする

王様　（0.7 , 0.6 , 0.2 , 0.7 ）
男　　（0.5 , 0.4 , 0.2 , 0.4 ）
女　　（0.1 , 0.1 , 0.4 , 0.3 ）
女王　（0.3 , 0.2 , 0.4 , 0.6 ）

上のベクトルを用いて“王様”ー“男”＋“女”を計算すると

分散表現(よく用いられる例)

王様　（0.7 , 0.6 , 0.2 , 0.7 ）
男　　（0.5 , 0.4 , 0.2 , 0.4 ）
女　　（0.1 , 0.1 , 0.4 , 0.3 ）
女王　（0.3 , 0.2 , 0.4 , 0.7 ）

(0.7-0.5+0.1 , 0.6-0.4+0.1 , 0.2-0.2+0.4 , 0.7-0.4+0.3)
=(0.3,0.3,0.4,0.6) ≒ 女王

文章のベクトル化について考える

文章のベクトル化

文章中の各単語の分散表現の平均を文章のベクトルとする

(例)「私は学生です」

各単語の分散表現
私　　(0.5 , 0.3 , 0.8 , 0.4)
は　　(0.2 , 0.7 , 0.2 , 0.6)
学生　(0.7 , 0.2 , 0.4 , 0.1)
です　(0.4 , 0.1 , 0.2 , 0.8)

(0.5 + 0.2 + 0.7 + 0.4 / 4 ,
  0.3 + 0.7 + 0.2 + 0.1 / 4 ,
  0.8 + 0.2 + 0.4 + 0.2 / 4 ,
  0.4 + 0.6 + 0.1 + 0.8 / 4)
= (0.45 , 0.325 , 0.4 , 0.475)
→「私は学生です」の分散表現

文章のベクトル化

どのように単語の分散表現を獲得するのか

Word2Vec

2013年に，トマス・ミコロフを中心としたGoogleの研究者らによって開発された，単語の分散表現を獲得する手法

Word2Vec


入力
テキストファイル
私 は 町田 に 住んで
います が 町田 は
良い 町 です
Word2Vec
獲得したいもの
単語の分散表現
[0.18, 0.23, 0.03,
-0.04 ,0.14, 0.06,
0.27, 0.44 ,0.41,
0.27, 0.63, -0.17]

Word2Vec(前処理)

入力される処理対象が日本語などの場合，分かち書き が必要

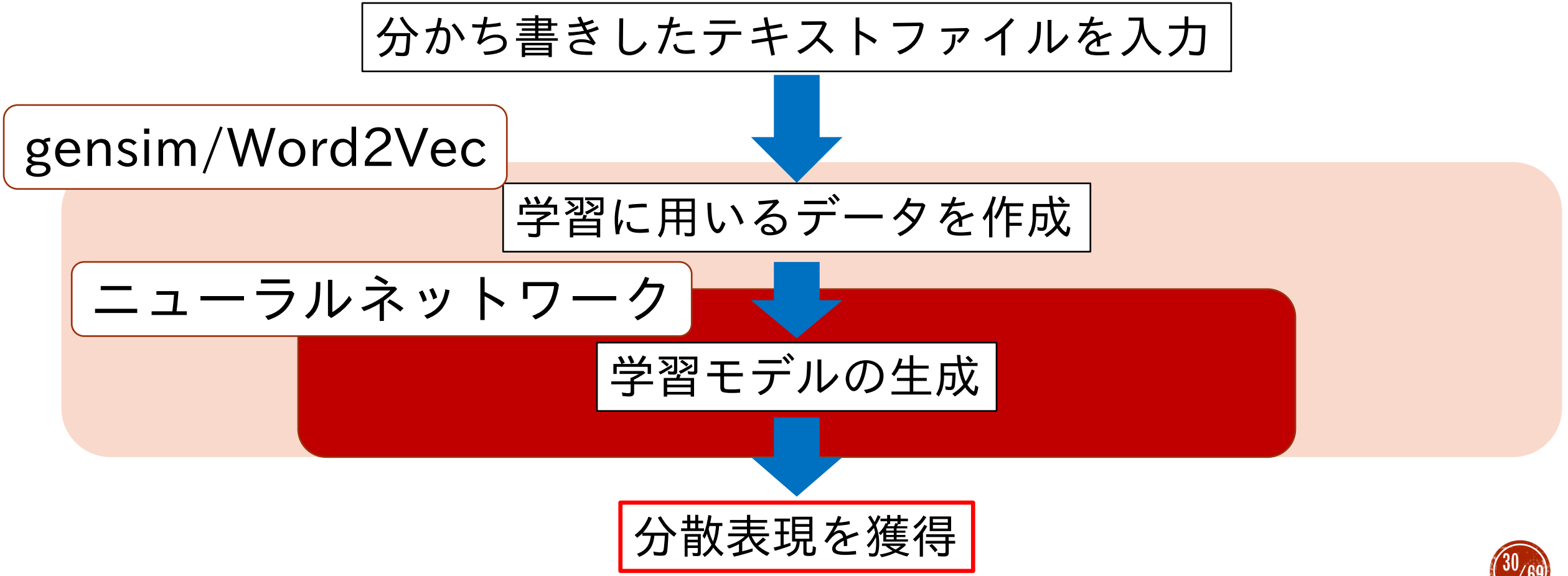
文章を単語ごとに分解
(例)　「 私は町田に住んでいますが町田は良い町です」

「 私　は　町田　に　住んで　います　が　町田　は　良い　町　です 」

Word2Vec

使用ライブラリ: gensim

Pythonのトピック分析ライブラリ
文書の解析などを行ってくれる
Word2Vecもgensimに実装されている



学習に用いるデータの作成

- 注目語と周辺単語の集合
- 重複のない単語集合(ボキャブラリ)

注目語と
周辺単語の集合

(注目語 , 周辺単語の集合)
(私 , {は , 町田})

注目語

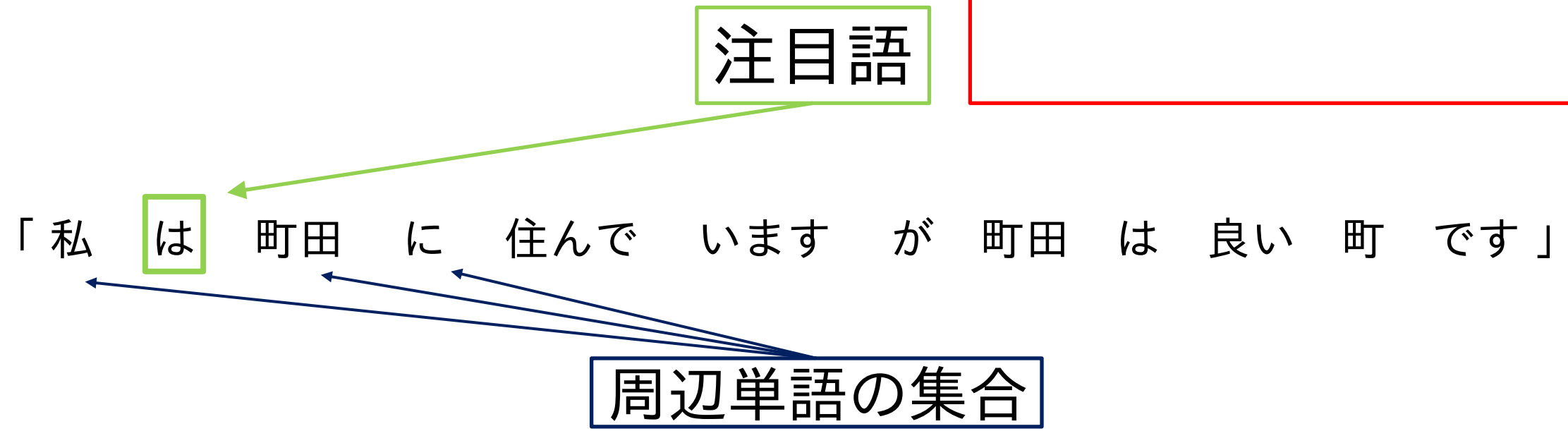
「私　は　町田　に　住んで　います　が　町田　は　良い　町　です」

周辺単語の集合

※周辺単語の範囲を２とした場合

注目語と
周辺単語の集合

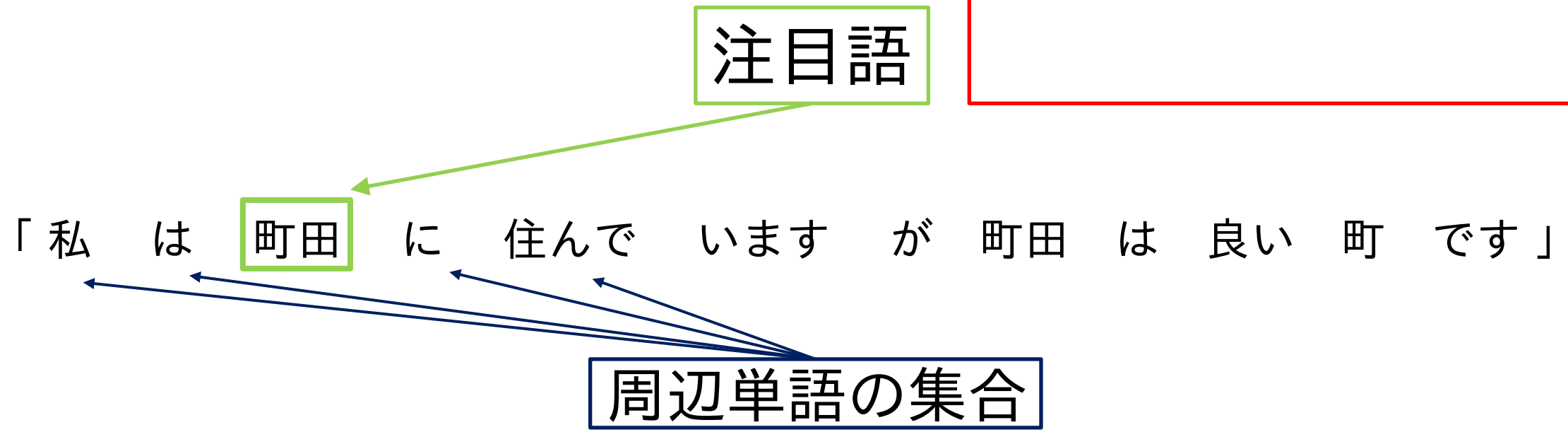
(注目語 , 周辺単語の集合)
(私 , {は , 町田})
(は , {私 , 町田 , に})



※周辺単語の範囲を２とした場合

注目語と
周辺単語の集合

(注目語 , 周辺単語の集合)
(私 , {は , 町田})
(は , {私 , 町田 , に})
(町田 , {私 , は , に , 住んで})

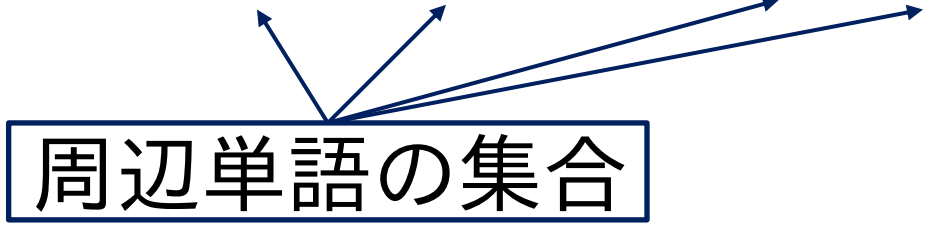


※周辺単語の範囲を２とした場合

注目語と
周辺単語の集合

(注目語 , 周辺単語の集合)
(私 , {は , 町田})
(は , {私 , 町田 , に})
(町田 , {私 , は , に , 住んで})
⋮
(町田 , {います , は , に , 良い})

「 私　は　町田　に　住んで　います　が　町田　は　良い　町　です 」



※周辺単語の範囲を 2 とした場合

注目語と
周辺単語の集合

(注目語 , 周辺単語の集合)
(私 , {は , 町田})
(は , {私 , 町田 , に})
(町田 , {私 , は , に , 住んで})
⋮
(町田 , {います , は , に , 良い})
⋮
(です , {良い , 町})

注目語

「 私　は　町田　に　住んで　います　が　町田　は　良い　町　です 」

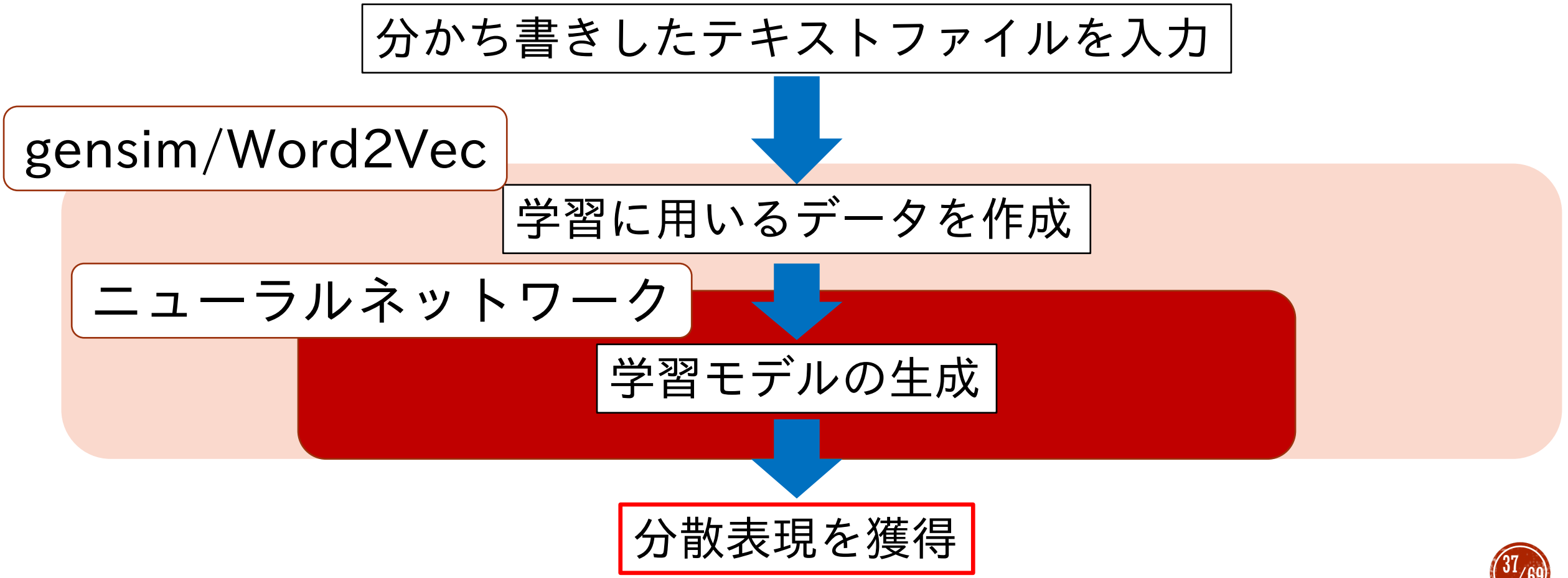
周辺単語の集合

※周辺単語の範囲を２とした場合

Word2Vec(再掲)

使用ライブラリ: gensim

Pythonのトピック分析ライブラリ
文書の解析などを行ってくれる
Word2Vecもgensimに実装されている



Word2Vec

Word2Vecでの学習モデルの生成にはニューラルネットワークを用いる

ボキャブラリ

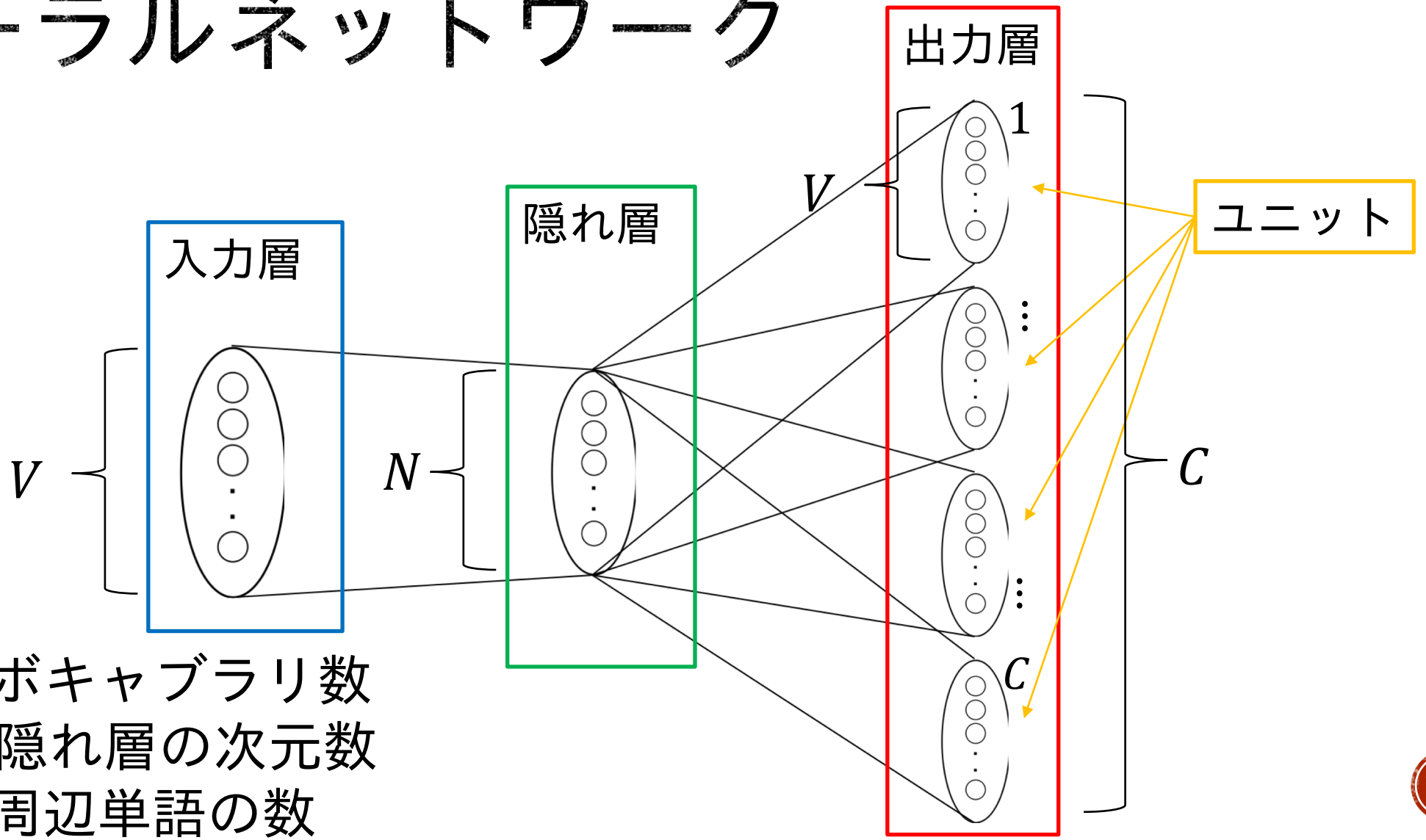
文章に出現する単語から重複のない単語集合

「私　は　町田　に　住んで　います　が　町田　は　良い　町　です」

語順は無視

{ います , が , 住んで , です , に , は , 町 , 町田 , 良い , 私 }

Word2Vecの
ニューラルネットワーク

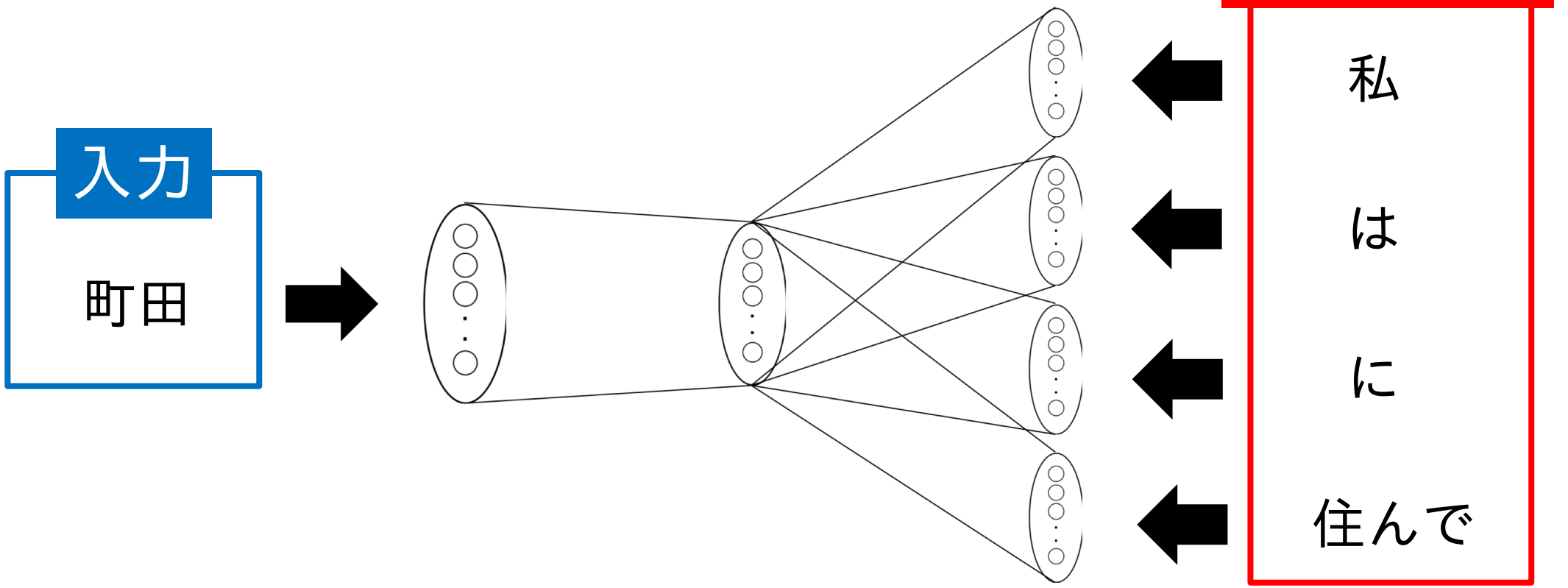


V =ボキャブラリ数
N =隠れ層の次元数
C =周辺単語の数

Word2Vecの
ニューラルネットワーク

(注目語 , 周辺単語の集合)
⋮
(町田 , {私 , は , に , 住んで})
⋮
(町田 , {います , は , に , 良い})
⋮

入力　　：注目語
正解データ　：入力した注目語の周辺単語の集合



Word2Vecの
ニューラルネットワーク

one – hot ベクトル

ある要素だけが 1 それ以外が 0 で表現されたベクトル

・入力においての注目語
・正解データにおいての周辺単語

} one – hot ベクトルを用いて表現

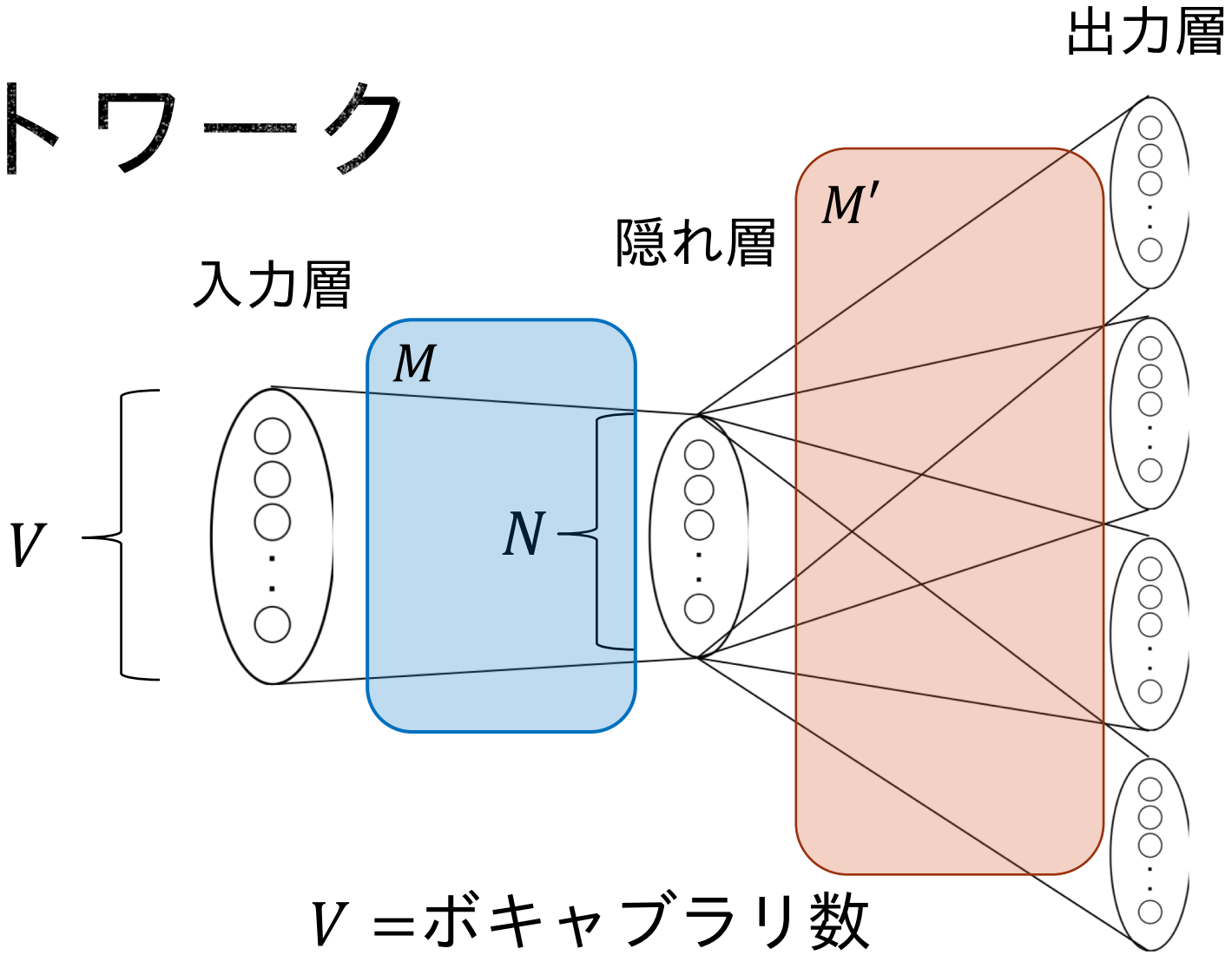
(例)
ボキャブラリ = { います , が , 住んで , です , に , は , 町 , 町田 , 良い , 私 }
町田　　=(0 , 0 , 0 , 0 , 0 , 0 , 0 , 1 , 0 , 0)

Word2Vecの
ニューラルネットワーク

入力層から隠れ層への
重み行列
$\rightarrow M \in \mathbb{R}^{V \times N}$

隠れ層から出力層への
重み行列
$\rightarrow M' \in \mathbb{R}^{N \times V}$

M'は各ユニットで共通

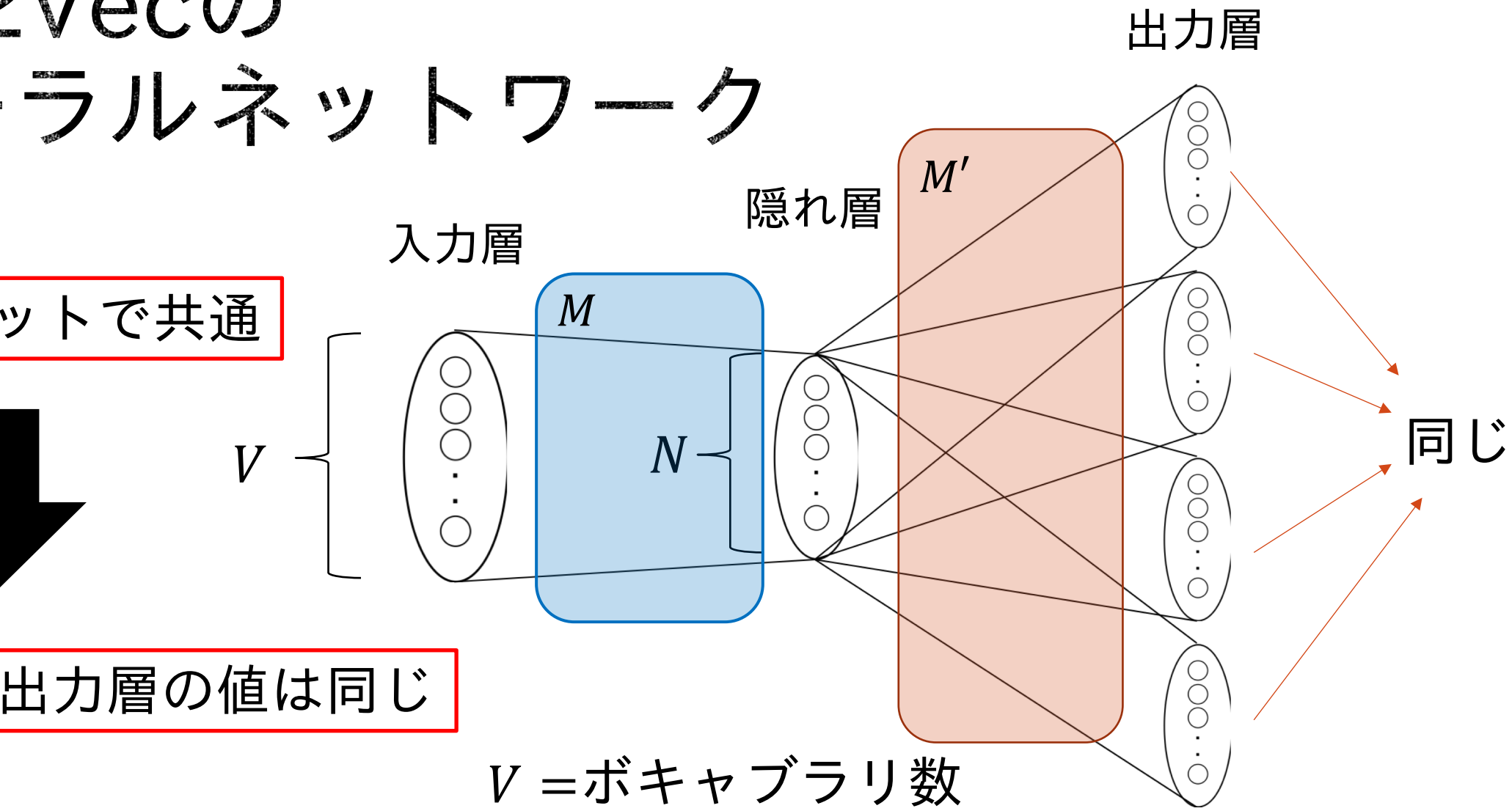


V ＝ボキャブラリ数
N ＝隠れ層の次元数

Word2Vecの
ニューラルネットワーク

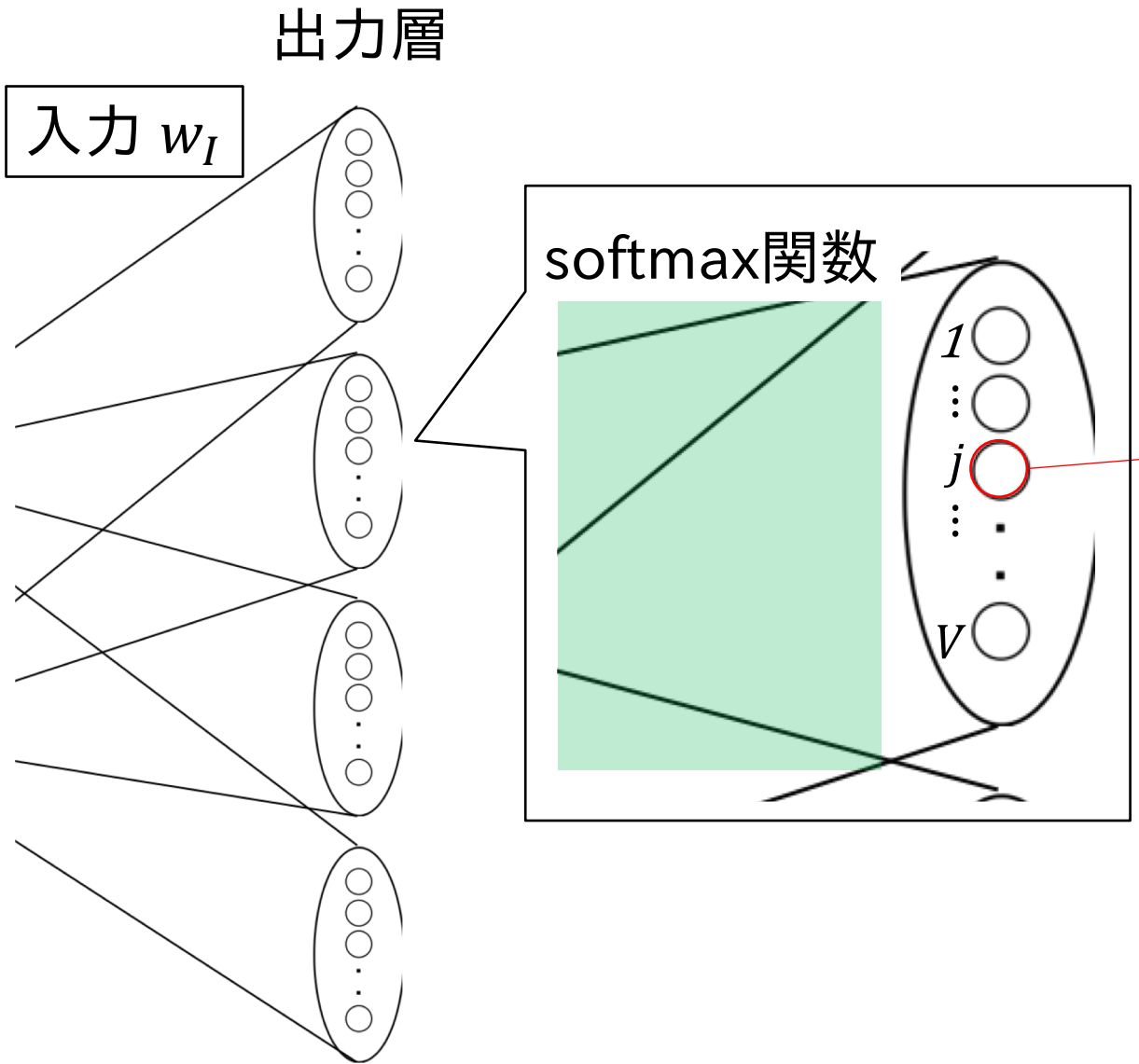
M' は各ユニットで共通

各ユニットの出力層の値は同じ



V ＝ボキャブラリ数
N ＝隠れ層の次元数

Word2Vecのニューラルネットワーク



入力 w_I の各ユニットの j 番目の出力:

w_j が w_I の周辺語となる条件付き確率

$$p(w_j|w_I)$$

を近似しているものと解釈

Word2Vecの
ニューラルネットワーク
(再掲)

(注目語 , 周辺単語の集合)

⋮

(町田 , {私 , は , に , 住んで})

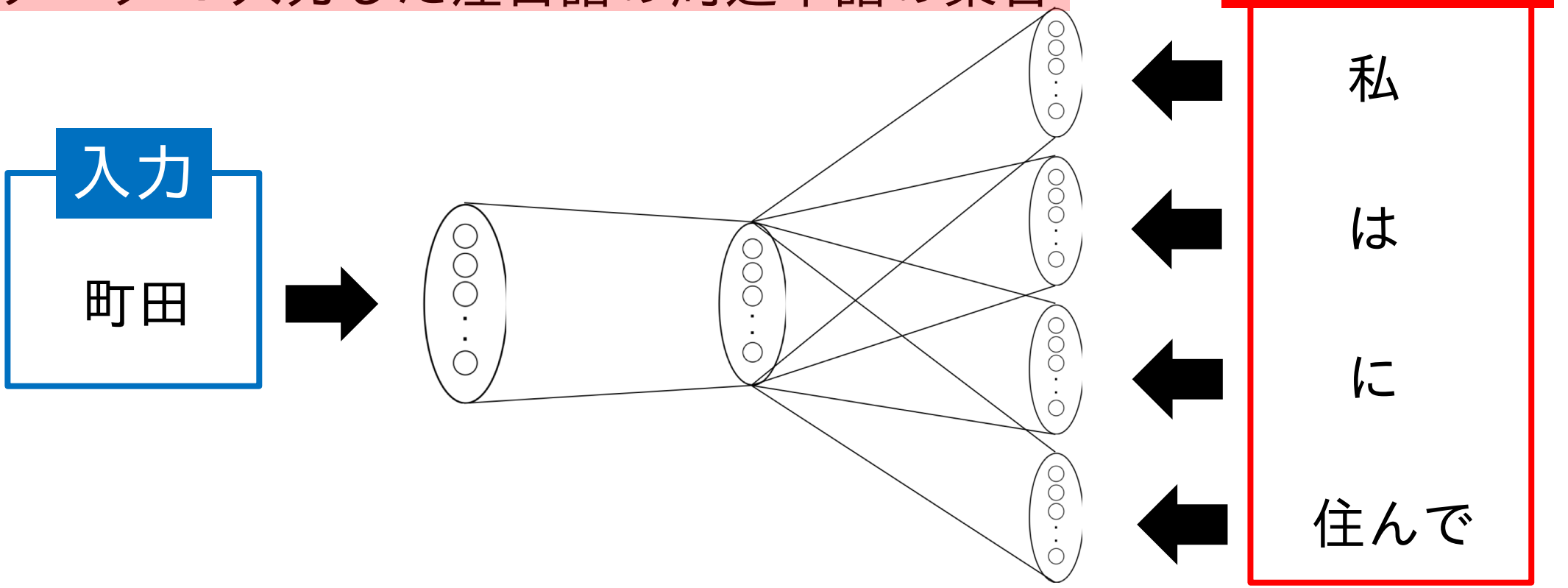
⋮

(町田 , {います , は , に , 良い})

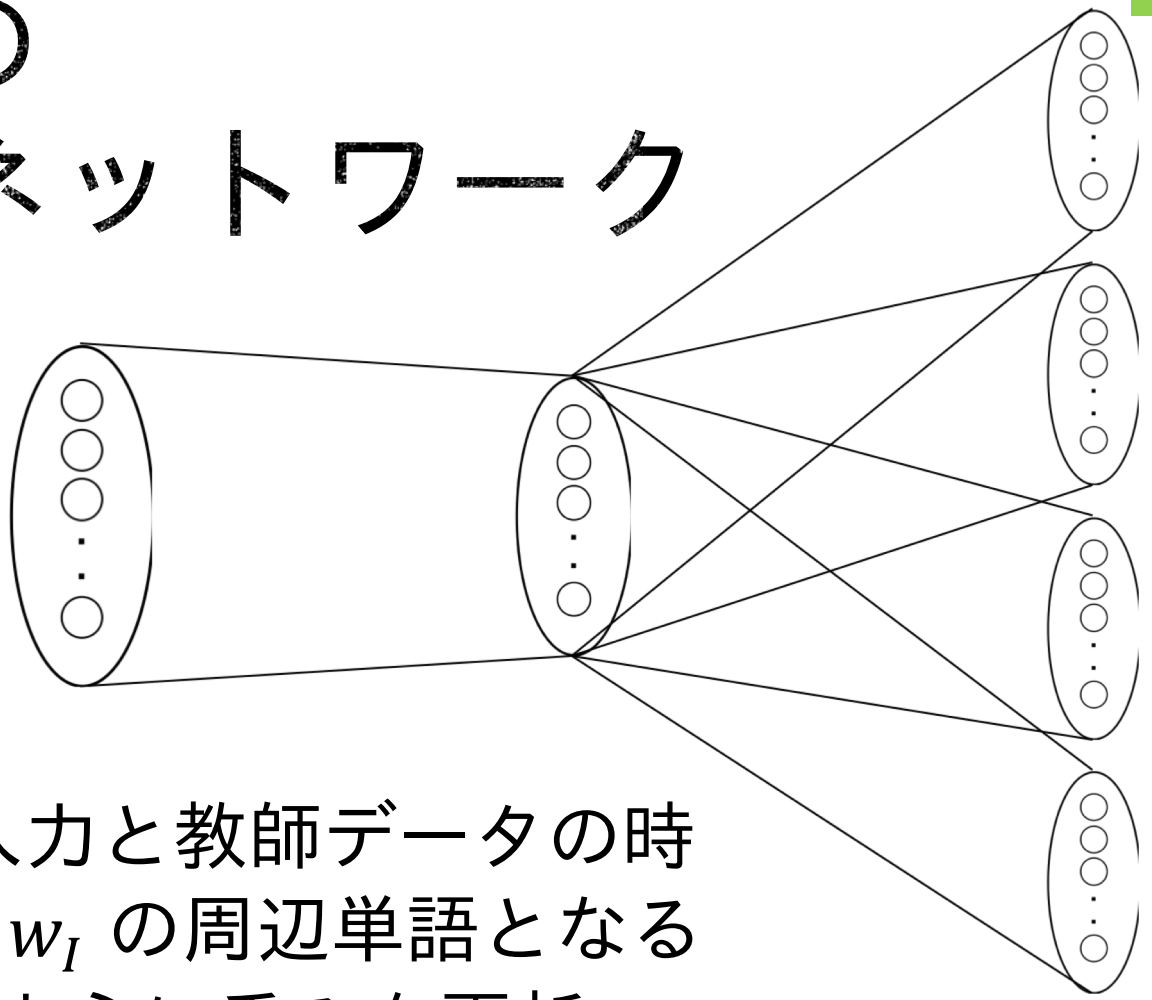
⋮

入力　　：注目語

正解データ　：入力した注目語の周辺単語の集合



Word2Vecの
ニューラルネットワーク

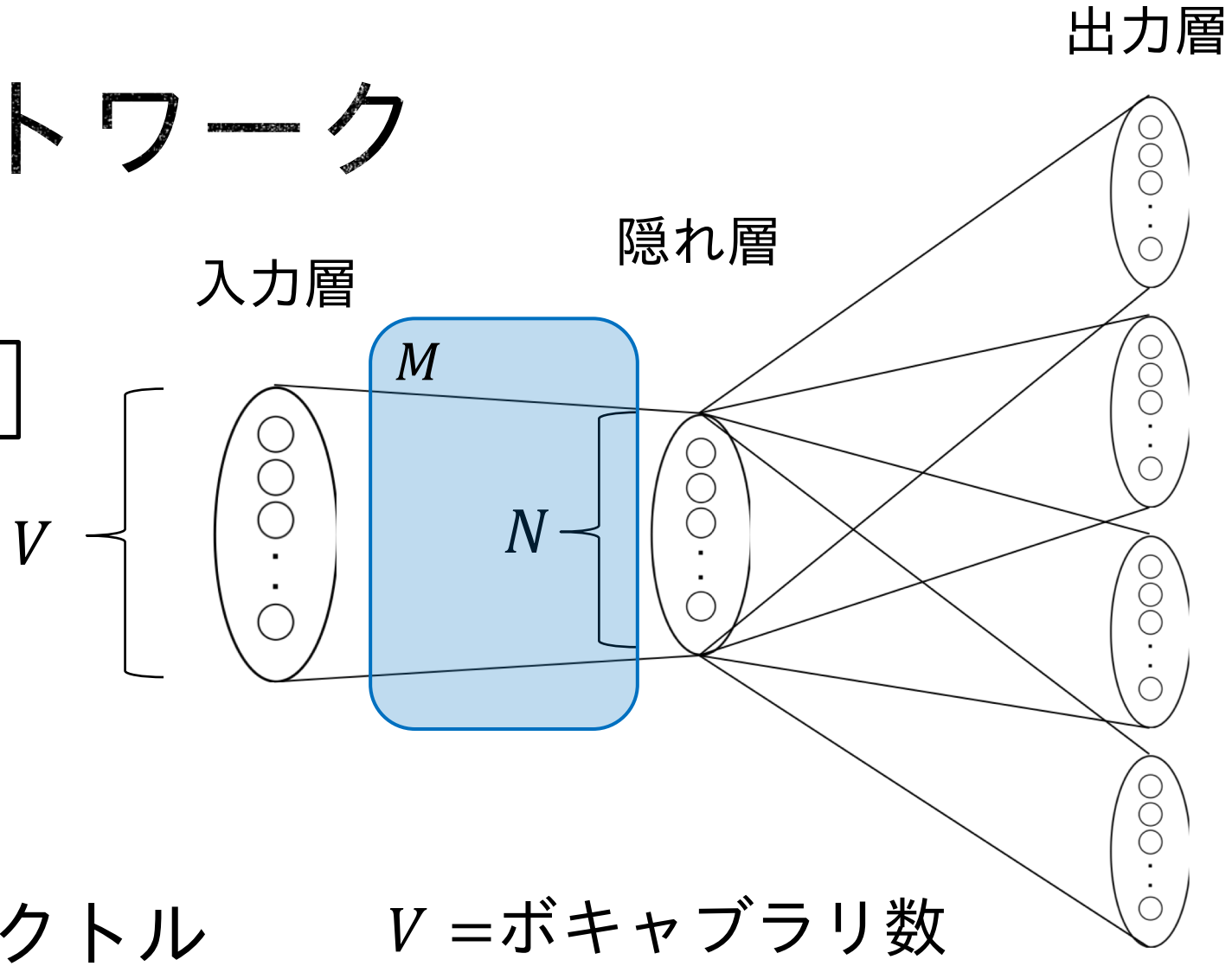


(w_I , $\{w_{O,1}, \cdots, w_{O,C}\}$) が入力と教師データの時
$(w_{O,1}, \cdots, w_{O,C})$ は同時に w_I の周辺単語となる
→同時確率が大きくなるように重みを更新

$$p(w_{O,1}, \cdots w_{O,C} | w_I)$$

Word2Vecの
ニューラルネットワーク

学習終了時の $(V \times N)$-行列 M



単語の分散表現を表す

行 i が w_i の分散表現ベクトル

V =ボキャブラリ数
N =隠れ層の次元数

演習の流れ(再掲)

Webページからデータの収集

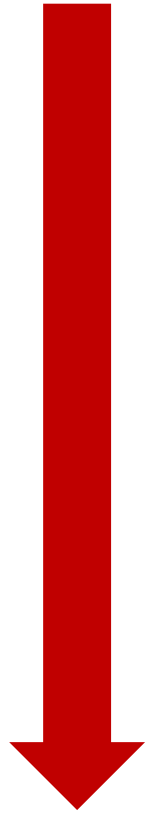
Word2Vecを用いて単語の分散表現を獲得

獲得した分散表現の平均ベクトルを用いて文章をベクトル化

文章同士の類似度を算出するプログラムの作成

演習内容

モデル生成に用いるデータの収集
データの前処理
学習モデルの構築
検証データの収集
検証データの前処理
~~検証~~



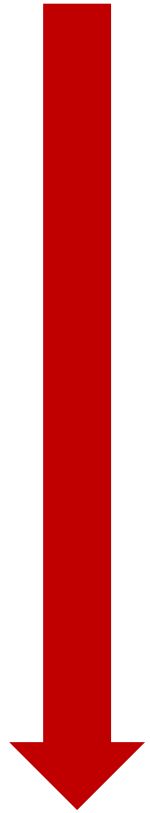
開発環境

言語	Python 3.6

ライブラリ	使用用途
MeCab 0.996	形態素解析エンジン
gensim 3.6.0	トピック分析ライブラリ
NumPy 1.14.5	数値計算ライブラリ
SciPy 0.18.1	

演習内容

モデル生成に用いるデータの収集
データの前処理
学習モデルの構築
検証データの収集
検証データの前処理
~~検証~~



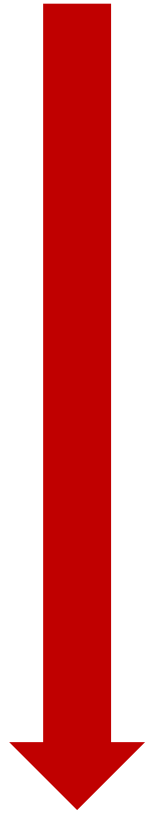
モデル生成に用いるデータの収集

Webページからデータをダウンロード
(https://dumps.wikimedia.org/jawiki/latest/jawiki-latest-pages-articles.xml.bz2)

モデル生成に用いるデータ：日本語のWikipedia(2018年11月15日時点)
データ形式：XML

演習内容

モデル生成に用いるデータの収集
データの前処理
学習モデルの構築
検証データの収集
検証データの前処理
~~検証~~



データの前処理

XMLデータをテキストファイルに変換

テキストファイルを分かち書き

データの前処理

XMLファイルをテキストファイルに変換
→学習に必要のない画像やテーブルなどを排除

使用したツール
WikiExtractor

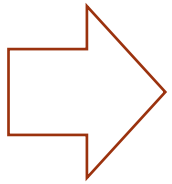
Wikipediaのデータを
テキストファイルに変換するPythonスクリプト

データの前処理

文章を単語ごとに分解し、Word2Vecに読み込める形式に変換
→分かち書き

使用したライブラリ
MeCab

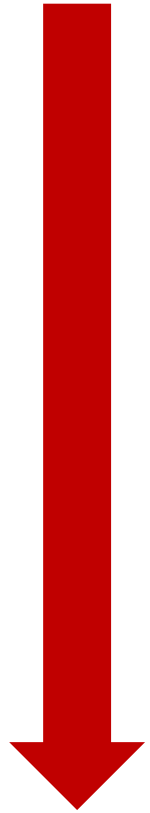
（言語学の用語に沿って）「動物のコミュニケーションの体系」も「言語」と呼ぶこともある、という主張がある。しかし、チョムスキー理論では「普遍文法」などの概念において、言語は人間のものという大前提があり、どういう意味で「言語学の用語に沿って」なのかは不明確である。



言語 学 の 用語 に 沿って ） 「 動物 の コミュニケーション の 体系 」 も 「 言語 」 と 呼ぶ こと も ある 、 という 主張 が ある 。 しかし 、 チョムスキー 理論 で は 「 普遍 文法 」 など の 概念 において 、 言語 は 人間 の もの という 大前提 が あり 、 どういう 意味 で 「 言語 学 の 用語 に 沿って 」 な のか は 不 明確 で ある 。

演習内容

モデル生成に用いるデータの収集
データの前処理
学習モデルの構築
検証データの収集
検証データの前処理
~~検証~~



学習モデルの構築

前処理したデータをWord2Vecに学習させる

入力：
分かち書きを行なったテキストファイル

コンテキストサイズ：10
単語ベクトルの次元数：50
反復回数：10

使用ライブラリ: gensim

Pythonのトピック分析ライブラリ
文書の解析などを行ってくれる
Word2Vecもgensimに実装されている

獲得した分散表現の例

```
>>> model['理科']
array([ 0.18544962,  0.23896538,  0.03860131, -0.04431016, -0.15026075,
        0.03993113, -0.17474075, -0.01082999, -0.21551247, -0.01959798,
        0.09714622, -0.01441354, -0.00405255,  0.10318583, -0.14130434,
       -0.06393439,  0.01948013,  0.02173181, -0.03315288,  0.14308907,
        0.25483203, -0.31765732,  0.13859855, -0.18632014,  0.03834172,
       -0.06380235, -0.16175115,  0.1489283 ,  0.19791387,  0.09809765,
        0.1433362 ,  0.16544716,  0.00195148, -0.05501816,  0.08512692,
       -0.1018239 , -0.10088514,  0.01144602, -0.07421104, -0.06121305,
       -0.32506856,  0.25344402,  0.14979304,  0.19982497, -0.14936675,
        0.13674489, -0.01170775, -0.14772697, -0.17484836, -0.01484798]
```

文章のベクトル表現

文章中の単語の分散表現を獲得し
その平均を文章のベクトルとする

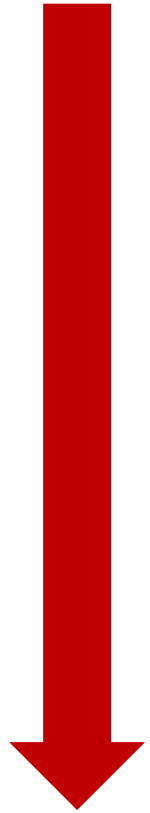
使用したツール
・MeCab
・NumPy

(例)「私は学生です」

```
[ 0.07986148  0.04527044  0.01589687 -0.06104406  0.04449122 -0.09763873
  0.0347955   0.04849926 -0.20341602 -0.13867146 -0.01858409 -0.04496396
 -0.00843627  0.01636032 -0.14355496  0.0311617  -0.08088029  0.0803988
 -0.08310359 -0.00273145  0.09813416  0.09200399  0.13857794  0.0280542
 -0.02865092 -0.09488252  0.08535072  0.004752    0.09277636  0.05131279
  0.16955565 -0.03171062 -0.05029654  0.01446473  0.01030451  0.04278848
  0.06675327  0.06284577  0.03440488 -0.15087609 -0.04545574  0.1410557
  0.15133661  0.02862334  0.01488443  0.04361499  0.05919865  0.15637758
  0.01051966  0.04230668]
```

演習内容

モデル生成に用いるデータの収集
データの前処理
学習モデルの構築
検証データの収集
検証データの前処理
~~検証~~



検証データの収集

インターネット上の小学生向け問題集から文章問題の解答を抜粋

(引用元：北九州市立教育センター
http://www.kita9.ed.jp/eductr/Handbook/Challengesheet/Elementaryschool/Challengesheet-s.html)

文章問題の解答群

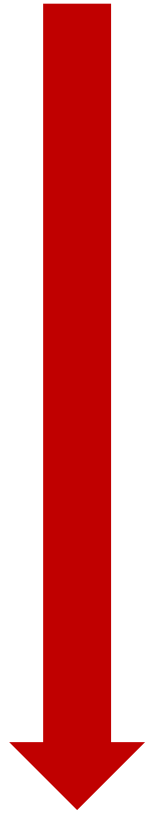
・磁石を近づけて，引きつけられるかどうかを調べる
・水に入れて，とけるかどうか調べる
・電気を通すか調べる

・成長するため
・土の養分を吸い上げるため
・体を支えるため

問題数：10
各問題の別解答数：約 4

演習内容

モデル生成に用いるデータの収集
データの前処理
学習モデルの構築
検証データの収集
検証データの前処理
~~検証~~



検証データの前処理

●漢字に直せる箇所を全て漢字に直す
　　ふやす⇨増やす
　　→学習モデルとしたデータに「ふやす」という語彙が存在しないため

●分かち書きを行う

類似度の算出

模範解答
食塩水の食塩は，蒸発しない

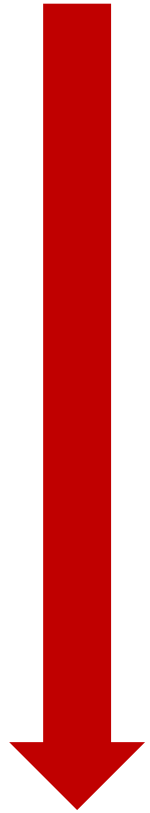
正しい別解(一部)
・食塩は，蒸発しない
0.9451318933299787
・食塩水の水は蒸発するが，食塩は蒸発しない
0.9649730467844014
・食塩水の水を蒸発させると，食塩は蒸発しない
0.9578893981449568

誤った解答(一部)

・水に入れて，とけるかどうか調べる
0.5663508519267949
・蒸発しない
0.6496236248444601

演習内容

モデル生成に用いるデータの収集
データの前処理
学習モデルの構築
検証データの収集
検証データの前処理
~~検証~~



まとめ

- 学習データに Wikipedia を用いて Word2Vec で分散表現を獲得
  →汎用性のある学習モデル/分散表現の取得
- 獲得した分散表現を用いて文章間の類似度を判定するプログラムを作成
- 理科の解答データの収集を行ったものの検証はできず

今後の課題

- 算出した類似度で正誤判定ができるのか検証
- 学習させるデータを教科書や参考書など分野に適した場合の検証
- Word2Vec は一般に対義語に弱いとされていることへの対応

Word2Vec

Word2Vec のアーキテクチャ(論理構造)は2種類存在する

- CBOW(Continuous Bag-of-words)
- Skip-gram

本演習で用いるのは Skip-gram

Skip-gram

「単語からその周辺単語を予測する」

(引用元:DeepAge, https://deepage.net/bigdata/machine_learning/2016/09/02/word2vec_power_of_word_vector.html，2019/1/24)

注目語に関連する単語が現れる確率を高くしていくモデル

Skip-gram

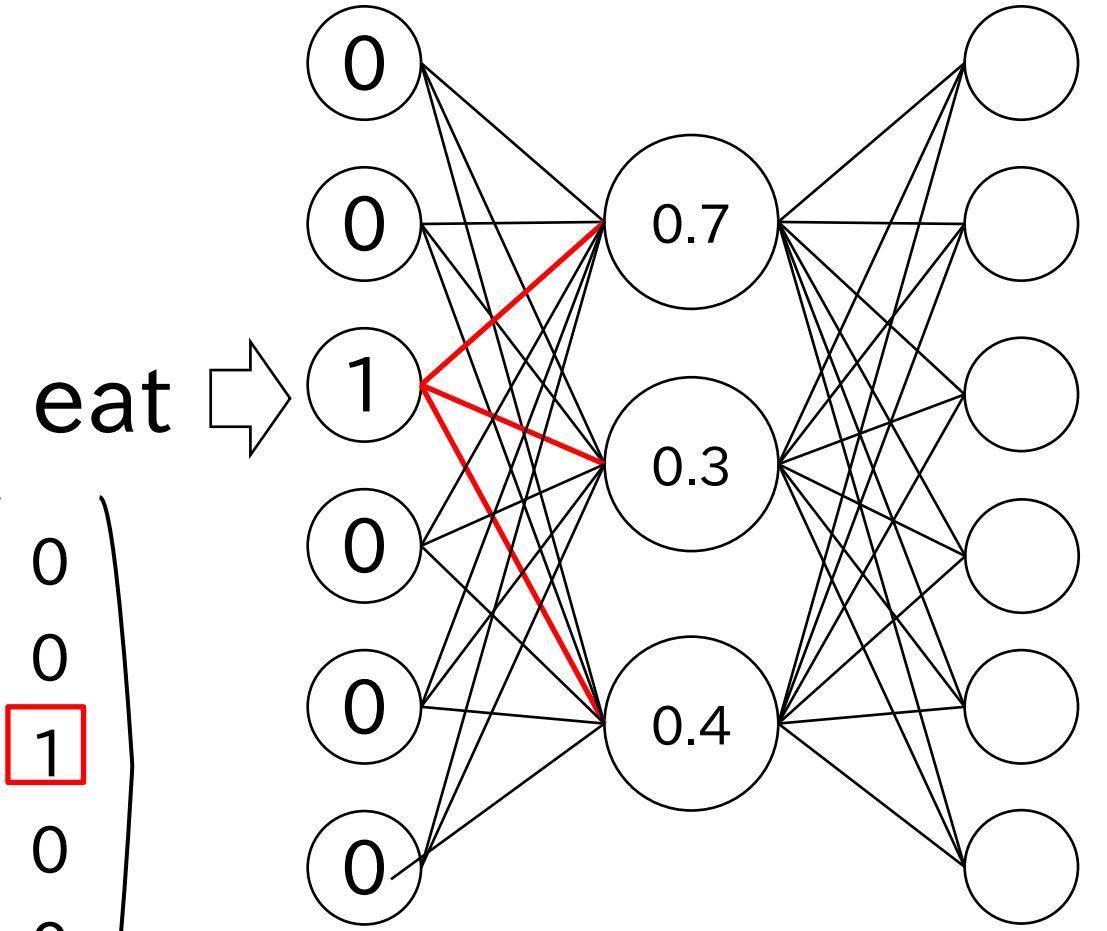
注目単語が与えられたときどんな単語が出現するか
という条件付き確率をsoftmax関数でモデル化

$$p(w_O|w_I) = \frac{exp\left(v'^{T}_{w_O} \cdot v_{w_I}\right)}{\sum_{w_v \in V} exp\left(v'^{T}_{w_v} \cdot v_{w_I}\right)}$$

w_O ＝周辺単語	v ＝入力ベクトル
w_I ＝注目単語	v' ＝出力ベクトル

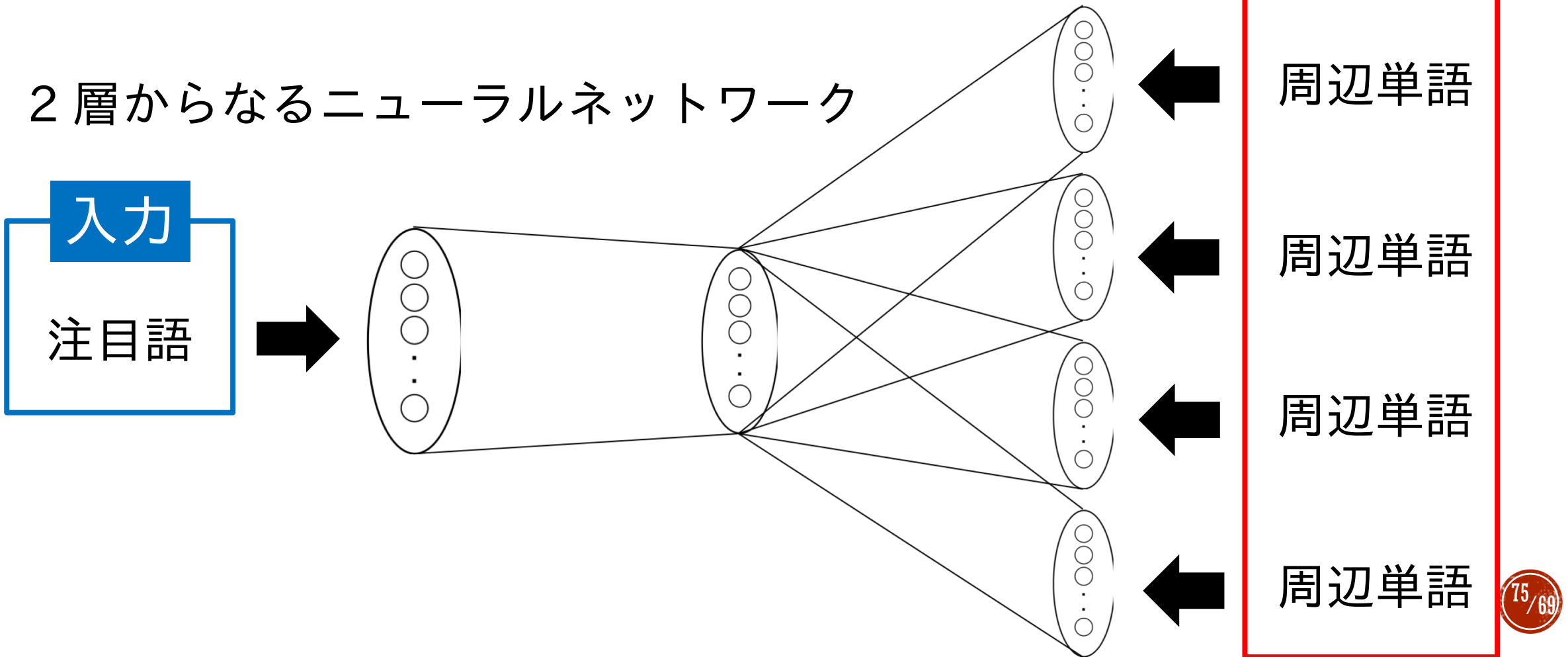
Word2Vec (隠れ層:h)

$$\begin{matrix} v_1 & v_2 & v_3 & v_4 & v_5 & v_6 \end{matrix}$$
$$\begin{bmatrix} 0.2 & 0.4 & 0.7 & 0.3 & 0.5 & 0.1 \\ 0.6 & 0.1 & 0.3 & 0.6 & 0.8 & 0.4 \\ 0.5 & 0.8 & 0.4 & 0.2 & 0.4 & 0.7 \end{bmatrix} \times \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$



Word2Vecの
ニューラルネットワーク

２層からなるニューラルネットワーク



Word2Vec

活性化関数としてsoftmax関数を用いるため
周辺単語１つずつは注目語が入力された時
その周辺単語が出現する確率(条件付き確率)とみなすことができる

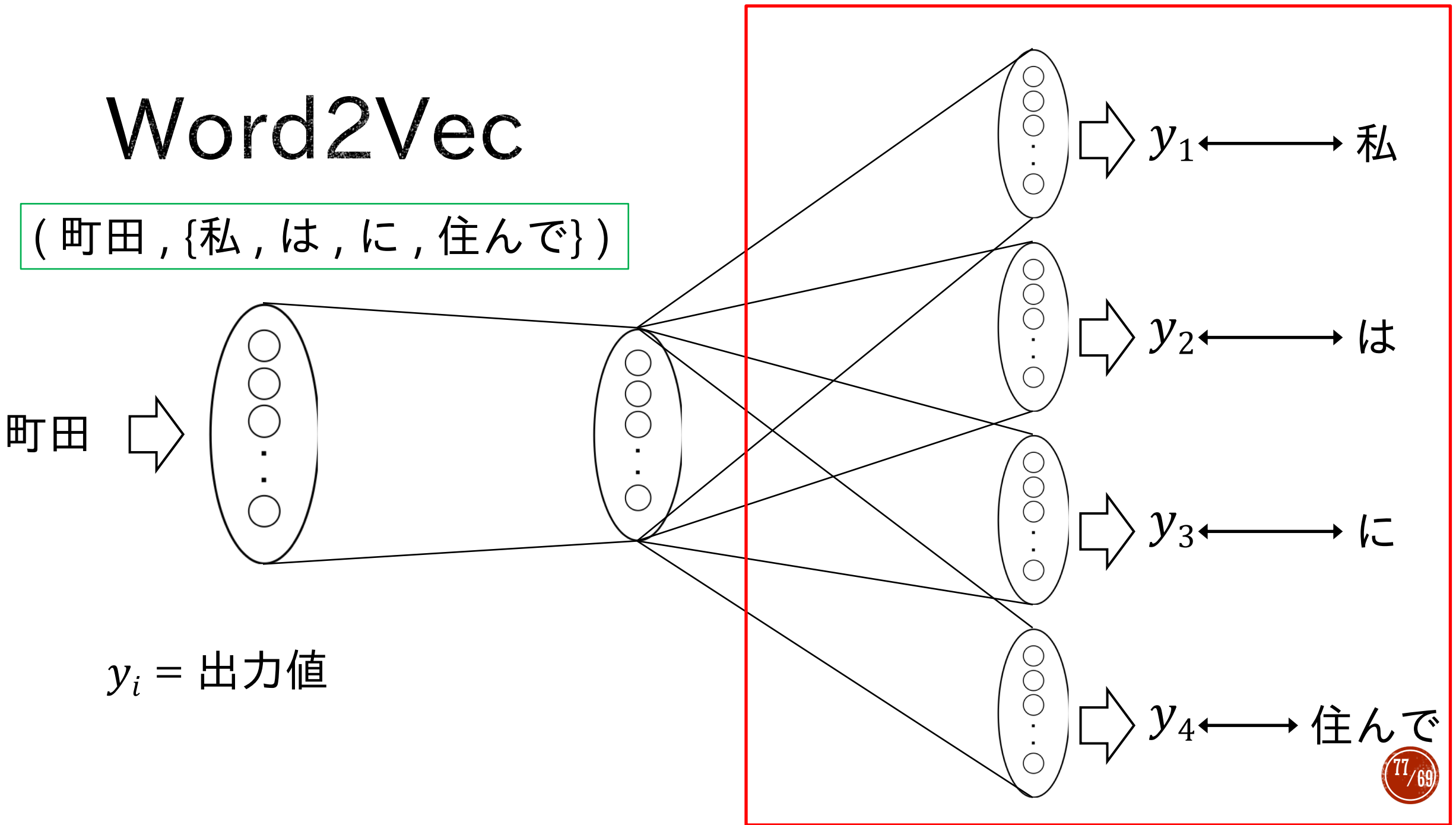
$$p(w_O|w_I) = \frac{exp\left(v'^{T}_{w_O} \cdot v_{w_I}\right)}{\sum_{w_v \in V} exp\left(v'^{T}_{w_v} \cdot v_{w_I}\right)}$$

w_O=私
w_I=町田

w_O =周辺単語　　v = 入力ベクトル
w_I =注目単語　　v' =出力ベクトル
V = ボキャブラリの単語数

Word2Vec

(町田 , {私 , は , に , 住んで})



y_i = 出力値

Word2Vec

周辺単語は前後のコンテキストサイズ分存在
注目語の周辺単語全てが同時に出現する確率(同時確率)を考える

$$p\left(w_{O,1}, \cdots w_{O,C} \middle| w_I\right) = \prod_{i=1}^{C} \frac{exp\left(v'^{T}_{w_{O,i}} \cdot v_{w_I}\right)}{\sum_{w_v \in V} exp\left(v'^{T}_{w_v} \cdot v_{w_I}\right)}$$

$w_{O,1}, \cdots w_{O,C}$ =
(私 , は , に , 住んで)
W_I = 町田

w_O =周辺単語　　v = 入力ベクトル
w_I =注目単語　　v' =出力ベクトル
V = ボキャブラリの単語数 C = 周辺単語の数

Word2Vec

$$p(w_{O,1}, \cdots w_{O,C} | w_I) = \prod_{i=1}^{C} \frac{exp\left(v'^{T}_{w_{O,i}} \cdot v_{w_I}\right)}{\sum_{w_v \in V} exp\left(v'^{T}_{w_v} \cdot v_{w_I}\right)}$$

同時確率が最大 → 単語の良い分散表現

注目語と周辺単語のデータセットを入力として学習

注目語と周辺単語のデータセットを入力
出力値をそれぞれの正解ラベル(周辺単語の one – hot ベクトル)と比較
その誤差を用いて重みを学習する
注目語の入力は one – hot ベクトル

単語	one – hot ベクトル
います	(1,0,0,0,0,0,0,0,0,0)
が	(0,1,0,0,0,0,0,0,0,0)
住んで	(0,0,1,0,0,0,0,0,0,0)
です	(0,0,0,1,0,0,0,0,0,0)
に	(0,0,0,0,1,0,0,0,0,0)
は	(0,0,0,0,0,1,0,0,0,0)
町	(0,0,0,0,0,0,1,0,0,0)
町田	(0,0,0,0,0,0,0,1,0,0)
良い	(0,0,0,0,0,0,0,0,1,0)
私	(0,0,0,0,0,0,0,0,0,1)

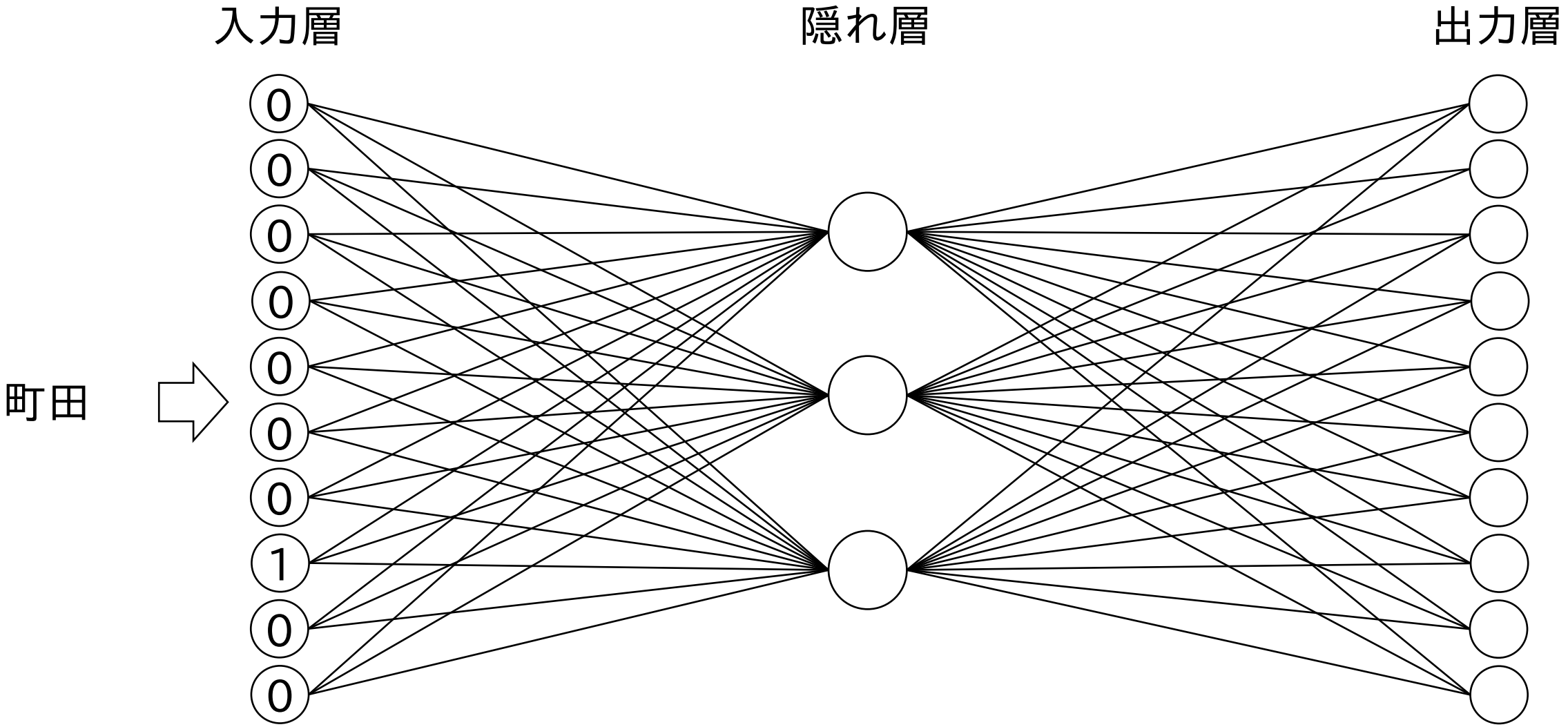
one – hot ベクトル
ある要素だけが 1 それ以外が 0 で
表現されたベクトルと単語を対応づける

Word2Vec(入力層:x)

入力
→ one – hot ベクトル
次元数
→V =ボキャブラリ数

V ={ います , が , 住んで , です , に , は , 町 , 町田 , 良い , 私 }
W_I(注目語) = 町田
(0 , 0 , 0 , 0 , 0 , 0 , 0 , 1 , 0 , 0)

Word2Vec(入力層:x)



Word2Vec (隠れ層:h)

重み

→ボキャブラリの全単語に対する単語ベクトルを横に並べた行列

$W = (\boldsymbol{v}_1, \boldsymbol{v}_2, \boldsymbol{v}_3, \boldsymbol{v}_4, \boldsymbol{v}_5, \boldsymbol{v}_6, \boldsymbol{v}_7, \boldsymbol{v}_8, \boldsymbol{v}_9, \boldsymbol{v}_{10})$

初期値

→ランダム

重み行列

→$W \in \mathbb{R}^{V \times N}$　(N = 単語ベクトルの次元数)

Word2Vec (隠れ層:h)

隠れ層の計算

$h = W\boldsymbol{x}$

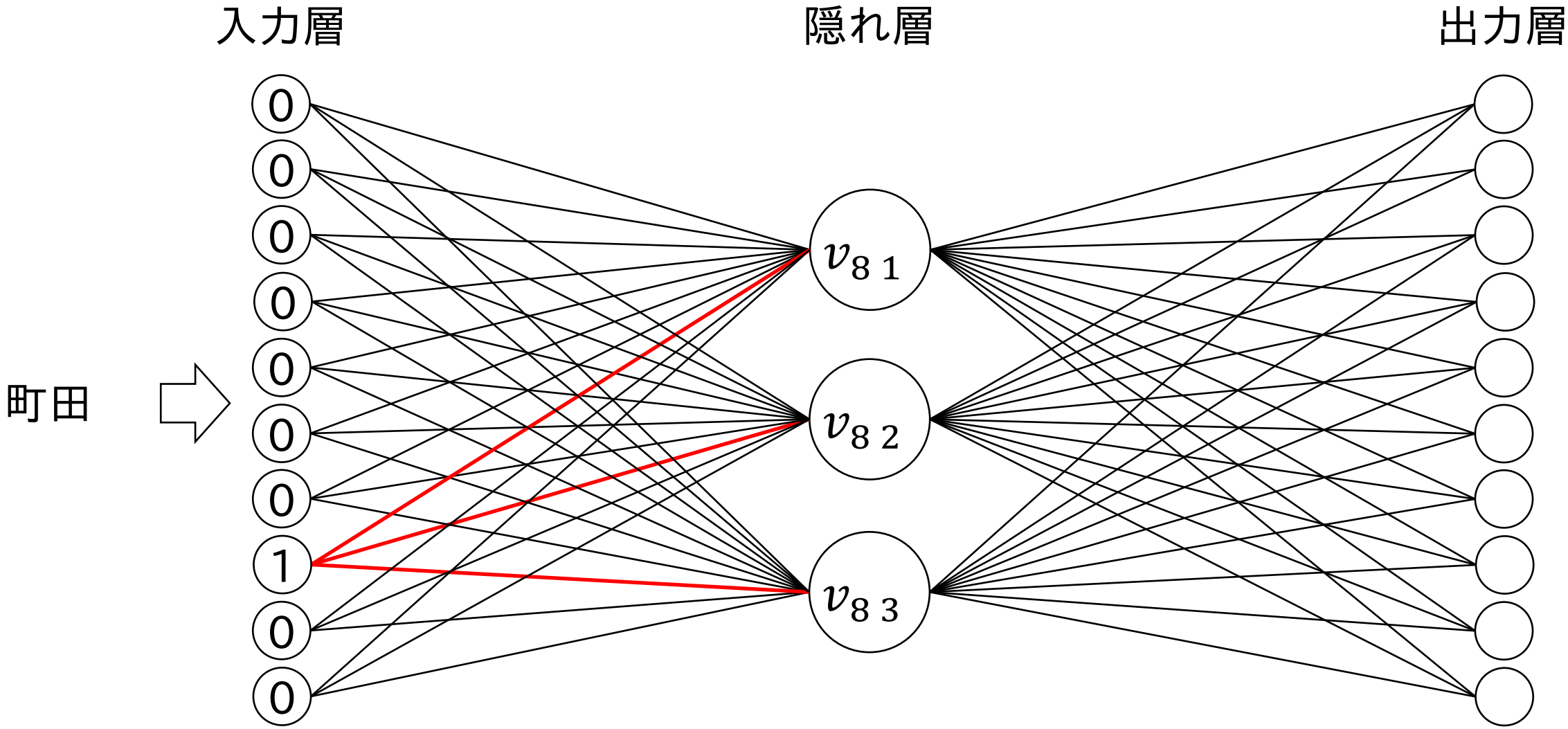
(例)　$N = 3$

$$\begin{matrix} \boldsymbol{v}_1 & \boldsymbol{v}_2 & \boldsymbol{v}_3 & \boldsymbol{v}_4 & \boldsymbol{v}_5 & \boldsymbol{v}_6 & \boldsymbol{v}_7 & \boldsymbol{v}_8 & \boldsymbol{v}_9 & \boldsymbol{v}_{10} \end{matrix}$$

$$\begin{bmatrix} v_{1\,1} & v_{2\,1} & v_{3\,1} & v_{4\,1} & v_{5\,1} & v_{6\,1} & v_{7\,1} & v_{8\,1} & v_{9\,1} & v_{10\,1} \\ v_{1\,2} & v_{2\,2} & v_{3\,2} & v_{4\,2} & v_{5\,2} & v_{6\,2} & v_{7\,2} & v_{8\,2} & v_{9\,2} & v_{10\,2} \\ v_{1\,3} & v_{2\,3} & v_{3\,3} & v_{4\,3} & v_{5\,3} & v_{6\,3} & v_{7\,3} & v_{8\,3} & v_{9\,3} & v_{10\,3} \end{bmatrix} \times \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \end{pmatrix} = \begin{matrix} \boldsymbol{v}_1 \\ \begin{bmatrix} v_{8\,1} \\ v_{8\,2} \\ v_{8\,3} \end{bmatrix} \end{matrix}$$

$$h = W x_{w_i} = \boldsymbol{v}_{w_i}$$

Word2Vec(隠れ層:h)



Word2Vec (出力層)

重み
→ボキャブラリの全単語に対する単語ベクトルを転置して、
　縦に並べた行列

$$W' = \begin{pmatrix} \boldsymbol{v}'^{T}_{1} \\ \boldsymbol{v}'^{T}_{2} \\ \boldsymbol{v}'^{T}_{3} \\ \boldsymbol{v}'^{T}_{4} \\ \boldsymbol{v}'^{T}_{5} \\ \boldsymbol{v}'^{T}_{6} \end{pmatrix}$$

重み行列
→$W' \in \mathbb{R}^{N \times V}$

活性化関数
→ softmax 関数

Word2Vec (出力層：u_C)

出力層の計算

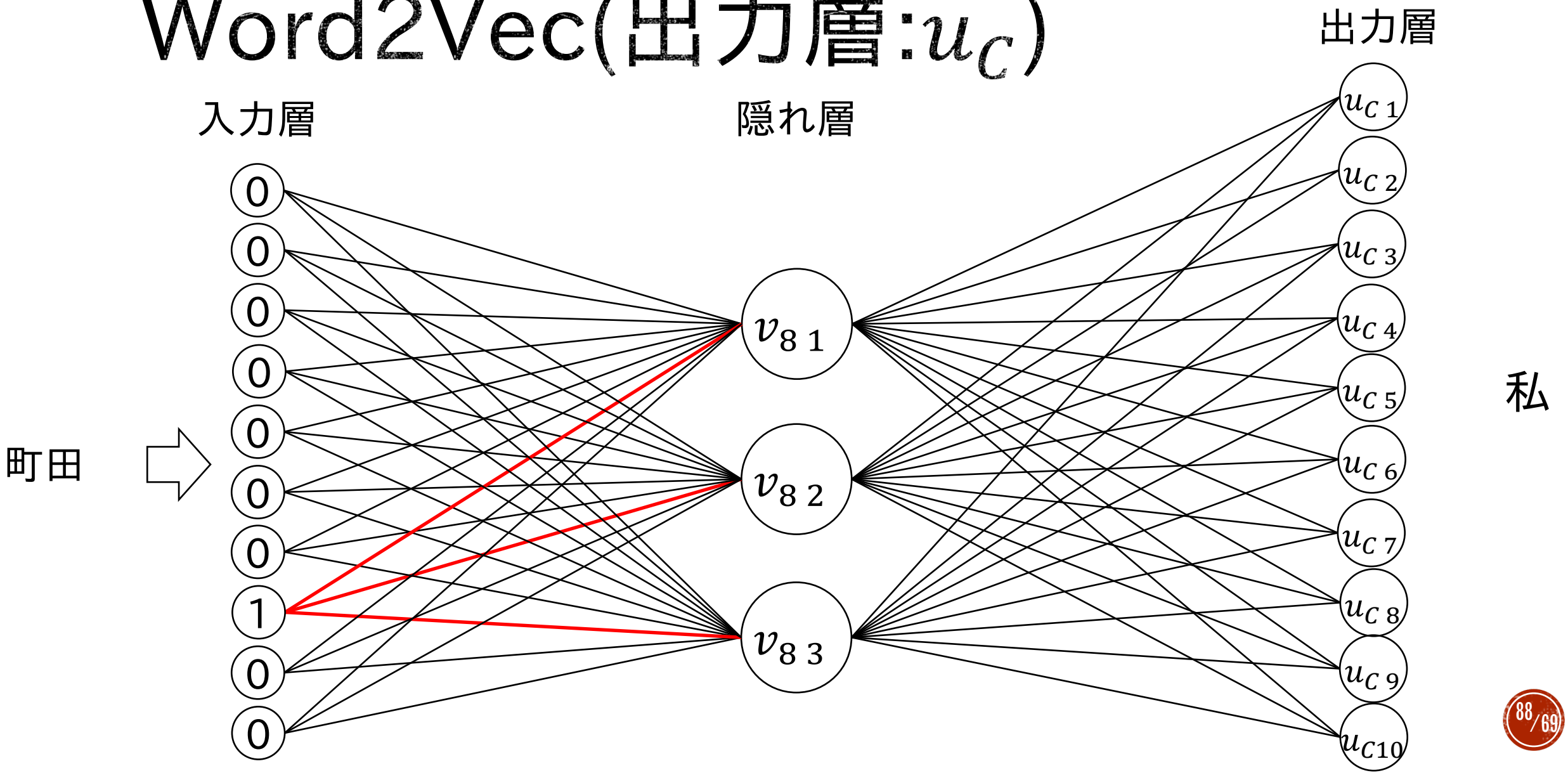
$u_{C\,i} = W' v_{w_I}$

C = 周辺単語

$$\begin{matrix} v'_1 \\ v'_2 \\ v'_3 \\ v'_4 \\ \vdots \\ v'_{10} \end{matrix} \begin{bmatrix} v'_{1\,1} & v'_{1\,2} & v'_{1\,3} \\ v'_{2\,1} & v'_{2\,2} & v'_{2\,3} \\ v'_{3\,1} & v'_{3\,2} & v'_{3\,3} \\ v'_{4\,1} & v'_{4\,2} & v'_{4\,3} \\ \vdots & \vdots & \vdots \\ v'_{10\,1} & v'_{10\,2} & v'_{10\,3} \end{bmatrix} \times \begin{bmatrix} v_{8\,1} \\ v_{8\,2} \\ v_{8\,3} \end{bmatrix} = \begin{bmatrix} u_{C\,1} \\ u_{C\,2} \\ u_{C\,3} \\ u_{C\,4} \\ \vdots \\ u_{C\,10} \end{bmatrix}$$

$$u_{C\,i} = v_i'^{T} \cdot v_{w_I}$$

Word2Vec(出力層:u_C)



Word2Vec(出力値:y)

出力された $u_{C\,i}$ にsoftmax関数をかけることで出力値 $y_{C\,i}$ が求まる
この $y_{C\,i}$ が注目語(w_I)に対する周辺単語(w_i)の条件付き確率

$$y_{C\,i} = \frac{exp(u_i)}{\sum_{v=1}^{V} exp(u_{c\,v})} = \frac{exp\left(v_i'^{T} \cdot v_{w_I}\right)}{\sum_{v=1}^{V} exp\left(v_v'^{T} \cdot v_{w_I}\right)} = p(w_i|w_I)$$

Word2Vec

$$p(w_{O,1}, \cdots w_{O,C} | w_I) = \prod_{i=1}^{C} \frac{exp\left(v'^{T}_{w_{O,i}} \cdot v_{w_I}\right)}{\sum_{w_v \in V} exp\left(v'^{T}_{w_v} \cdot v_{w_I}\right)}$$

同時確率を用いて重みを更新していく

Word2Vec

学習後の入力層から隠れ層への重みが
単語の分散表現となる

$$\begin{matrix} v_1 & v_2 & v_3 & v_4 & v_5 & v_6 & v_7 & v_8 & v_9 & v_{10} \end{matrix}$$
$$\begin{bmatrix} v_{1\,1} & v_{2\,1} & v_{3\,1} & v_{4\,1} & v_{5\,1} & v_{6\,1} & v_{7\,1} & v_{8\,1} & v_{9\,1} & v_{10\,1} \\ v_{1\,2} & v_{2\,2} & v_{3\,2} & v_{4\,2} & v_{5\,2} & v_{6\,2} & v_{7\,2} & v_{8\,2} & v_{9\,2} & v_{10\,2} \\ v_{1\,3} & v_{2\,3} & v_{3\,3} & v_{4\,3} & v_{5\,3} & v_{6\,3} & v_{7\,3} & v_{8\,3} & v_{9\,3} & v_{10\,3} \end{bmatrix} \times \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \end{pmatrix} = \begin{matrix} v_1 \\ \begin{bmatrix} v_{8\,1} \\ v_{8\,2} \\ v_{8\,3} \end{bmatrix} \end{matrix}$$