メディアプログラミング入門

第５回：トピック分析とWord Embedding

火５@本郷 2019年7月9日

情報理工学系研究科 数理・情報教育研究センター
准教授 山肩 洋子

第４回の課題：
夏目漱石の「吾輩は猫である」っぽい文を生成してみよう！

課題１：分かち書きの生成

- *text/wagahaiwa_nekodearu_org.txt*には、夏目漱石の小説「吾輩は猫である」の本文が記録されている
- このテキストを分かち書き文に変換して、*text/wagahaiwa_nekodearu_wakati.txt*に保存
- 出力ファイルの最初の1行を出力し、その状態で保存して提出

課題２：統計的単語tri-gramによるランダム文生成

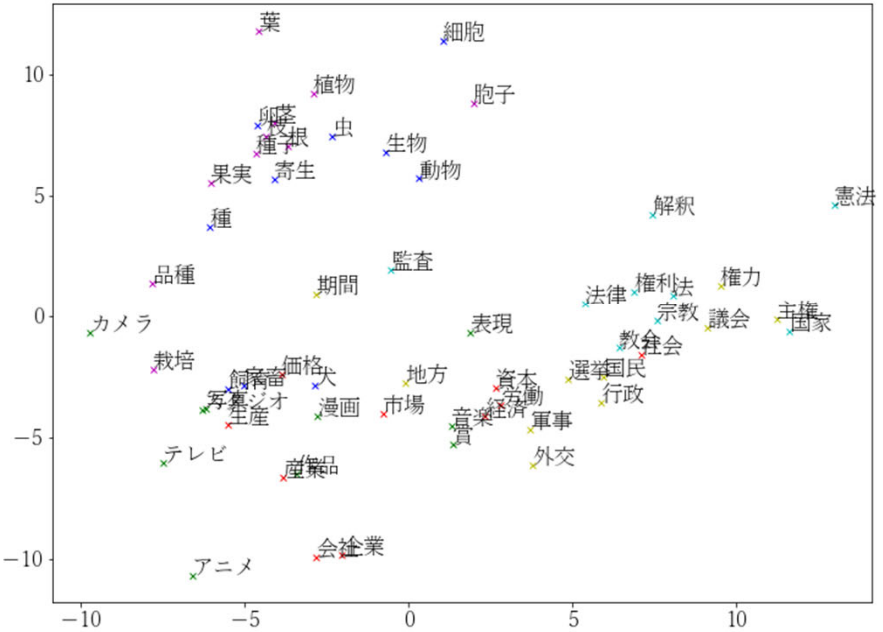
- 課題１で作成した*text/wagahaiwa_nekodearu_wakati.txt*を入力として、統計的tri-gramモデルを学習してください
- 学習モデルの変数名は*lm*とします
- また、2つ下のセルで、そのモデルを使って夏目漱石の小説「吾輩は猫である」っぽい文をランダムに生成してください
  - 生成文は「吾輩は」から始めて、「。」あるいは「」」で終わる一文です
  - 生成した文が出力した状態でこのファイルを保存して、提出してください

第5回 テキスト解析2：自分の文章の特徴を調べよう

講義内容：Pythonにおける文書の取り扱いや解析、検索方法について学習する

演習内容：自然言語処理ライブラリgensimを用いて、特徴語辞書作成やtf-idf、トピック分析 (LDA)、word2vecを試行する

Wikipediaの記事から重要語を抽出し、タグクラウドを生成 （上図はWikipediaカテゴリ 「動物」のタグクラウド）



- Newral networkの意味解析手法であるWord2vecによる単語マッピング
- 語彙をベクトル表現に変換すれば、機械学習に利用できる

本日の学習内容

- テキストのベクトル空間モデル
  - 単語のベクトル表現
  - 文書のベクトル表現
  - 文書のトピック分析の前処理・ストップワード
- トピック分析
  - 教師あり文書分類：tf-idf値による重要語抽出
  - 教師なし文書分類：潜在的意味解析
    （LDA：Latent Dirichlet Allocation）
  - 演習）Wikipedia記事のトピック分析
- Word Embedding
  - 深層学習による単語の分散表現生成：Word2Vec
  - 演習）gensimによるWord2vecの学習と可視化

テキストのベクトル空間モデル

テキストを機械学習で扱うには？

例えば…

- ある会社の株価が上がるか下がるかは、その会社のTwitterアカウントの文書が影響するかも！？
  → 株価の予測モデルにTwitterの情報も組み込みたい！
- 一般的な機械学習は数値のバラツキをモデル化する
  → テキストを数値に変換する必要がある

例）2019/4/10 @UTokyo_News
Tweet0 = “【イベント】GO GLOBAL 東大留学フェア2019、4月20日@駒場キャンパスにて開催！”

どうやって変換しましょう？

ベクトル表現に変換すれば学習できる
Tweet0 = [0.53, 0.45, 0.36, 0.26, 0.436, 0. 463,…]

何はともあれ単語分割

例）2019/4/10 @UTokyo_News
Tweet0 = “【イベント】GO GLOBAL 東大留学フェア2019、4月20日@駒場キャンパスにて開催！”

Janomeなどの形態素解析器

Tweet0_words＝ ['【', 'イベント', '】', 'GO GLOBAL', ' ', '東大', '留学フェア', '2019', '、', '4', '月', '20', '日', '@', '駒場キャンパス', 'にて', '開催', '！']

文書は単語の集合とみなす：Bag-Of-Words

- 語順は考慮しない（しなくてもだいたいわかる）
  - 「イベント」→イベント関係の記事？
  - 「開催」→開催案内らしい…
- その文書に特定の単語が登場するかしないか？

辞書と単語のベクトル表現

語彙 (Vocabulary)：ある文書集合に現れるすべての単語

- 前回の演習で、宮沢賢治の文書集合の語彙サイズは22,793語
- 語彙にはIDが付いている→これを使っていい？ダメ！

語彙リスト

ID	表記
…	…
7727	カブ
7728	カマ
7729	カムチャッカ
7730	**カムチヤツカ**
7731	**カムパネルラ**
7732	**カメレオン**
7733	カラ
7734	カラア
7735	カラカラ
…	…

「カムパネルラ」よりも「カムチヤツカ」のほうが大きい？
「カメレオン」と「カムチヤツカ」の平均は「カムパネルラ」！？

- ただの識別子であって大小関係を意味しない！
- しかし機械学習では大小関係を学習してしまう

one-hotベクトル

one-hotベクトル

- ベクトルの要素数＝語彙サイズ
- ある単語のベクトル表現は、その単語に該当するアドレスの値だけが1となり、残りが0

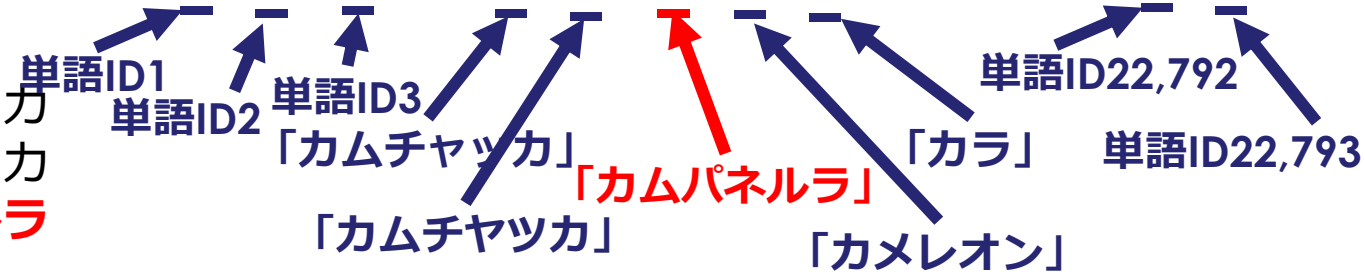
語彙リスト

ID	表記
…	…
7727	カブ
7728	カマ
7729	カムチャッカ
7730	カムチヤツカ
7731	**カムパネルラ**
7732	カメレオン
7733	カラ
7734	カラア
7735	カラカラ
…	…

例）「カムパネルラ」のone-hotベクトル

Word_vec['カムパネルラ']

=[0, 0, 0, …. 0, 0, 1, 0, 0, ……., 0, 0]



22,793次元のベクトルのうち、「カムパネルラ」に対応する7731番目の要素だけが1で、それ以外はすべて0

one-hotベクトルはテキストに限らずよく使われます
例）出身都道府県をベクトル表現に変換する

one-hotベクトルの使いどころ

アンケートの回帰分析

例）所属学部から得意科目を予測しよう

	好きな科目	所属学部
東大花子	社会	工学部
本郷太郎	国語	文学部
駒場次郎	数学	理学部

科目ID: 算数=1, 国語=2, 理科=3, 社会=4, 英語=5,…

学習用データ：

4→工学部

2→文学部

1→理学部

で回帰モデルを学習すればいい！

one-hotベクトルの使いどころ

アンケートの回帰分析

例）所属学部から得意科目を予測しよう

	好きな科目	所属学部
東大花子	社会	工学部
本郷太郎	国語	文学部
駒場次郎	数学	理学部

科目ID: 算数=1, 国語=2, 理科=3, 社会=4, 英語=5,…

学習用データ：

[0, 0, 0, 1, 0,…,0] →工学部

[0, 1, 0, 0, 0,…,0] →文学部

[1, 0, 0, 0, 0,…,0] →理学部

というようにone-hotベクトルにするべき！

文書の類似度計算

目的：ある文書 $\boldsymbol{d}_{\mathrm{A}}$ と $\boldsymbol{d}_{\mathrm{B}}$ がどの程度似ているかを評価したい

- $\boldsymbol{d}_{\mathrm{A}}$ と $\boldsymbol{d}_{\mathrm{B}}$ 以外にもたくさんの文書が与えられているとする。その文書集合$\boldsymbol{D}$における語彙を $V=[v_1, v_2, \ldots, v_{|V|}]$ とする
- 単語 v_i のone-hotベクトルの次元は $|V|$ (=語彙数)
- 文書 $\boldsymbol{d}$ の特徴ベクトル $\boldsymbol{d}=[d_1, d_2, d_3, \ldots, d_{|V|}]$
  - 次元は$|V|$
  - ある要素d_1は単語 $v_i \in V$ のその文書における重要度
  - 重要度とは？
    - その文書における単語 v_i の出現回数
      →つまり$\boldsymbol{d}$ は、その文書に含まれるすべての単語のone-hotベクトルの合計
    - TF-IDFやLDAで求めた値とすることもできる
      （後で出てきます）

演習：短い文書間の類似度を計算してみよう

文書（8種類）

'私 は 本 を 読む 。'
'今日 は 晴天 だ 。'
'私 は 私 だ 。'
'本 は 本 で 本 だ 。'
'私 は 本 を 読む 。'
'私 本 を 読む 。‘
'私 は 本 を 読む 。 本 を 私 は 読む 。'
'明日 雨 が 降る ！'

語彙リスト

ID	単語
0	。
1	が
2	だ
3	で
4	は

ID	単語
5	を
6	今日
7	明日
8	晴天
9	本

ID	単語
10	私
11	読む
12	降る
13	雨
14	！

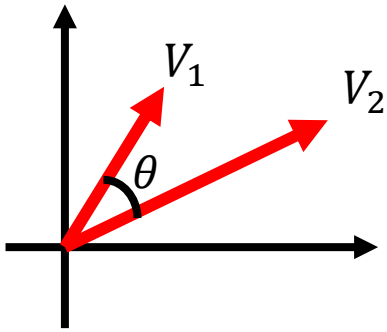
文書	。	が	だ	で	は	を	今日	明日	晴天	本	私	読む	降る	雨	！
「私 は 本 を 読む 。」	1	0	0	0	1	1	0	0	0	1	1	1	0	0	0
「今日 は 晴天 だ 。」	1	0	1	0	1	0	1	0	1	0	0	0	0	0	0
「私 は 私 だ 。」	1	0	1	0	1	0	0	0	0	0	2	0	0	0	0
「本は 本 で 本 だ 。」	1	0	1	1	1	0	0	0	0	3	0	0	0	0	0
「私 は 本 を 読む 。」	1	0	0	0	1	1	0	0	0	1	1	1	0	0	0
「私 本 を 読む 。」	1	0	0	0	0	1	0	0	0	1	1	1	0	0	0
「私 は 本 を 読む 。 本 を 私 は 読む 。」	2	0	0	0	2	2	0	0	0	2	2	2	0	0	0
「明日 雨 が 降る ！」	0	1	0	0	0	0	0	1	0	0	0	0	1	1	1

コサイン類似度 (cosine similarity)

- 文書間の類似度は、この文書ベクトル同士がどれだけ似通っているかで判断することができる
- ベクトル間の類似度の尺度はいろいろありますが、中でもよくつかわれるのが**コサイン類似度**

ベクトル V_1 とベクトル V_2 の類似度:

$$\cos\_\mathrm{sim}(V_1 V_2) = \cos\theta = \frac{V_1 \cdot V_2}{|V_1||V_2|}$$



ベクトルの大きさ（＝文書の長さ）は関係ない！

「私 は 本 を 読む 。」の文書ベクトルは [1, 0, 0, 0, 1, 1, 0, 0, 0, 1, 1, 1, 0, 0, 0]
「私 は 私 だ 。」の文書ベクトルは [1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 2, 0, 0, 0, 0]

$V_1 \cdot V_2$
=1x1 + 0x0 + 0x1 + 0x0 + 1x1 + 1x0 + 0x0 + 0x0 + 0x0 + 1x0 + 1x2 + 1x0 + 0x0 + 0x0 + 0x0
=4

$|V_1|$ =sqrt(1+1+1+1+1+1) = sqrt(6), $|V_2|$ =sqrt(1+1+1+2**2) = sqrt(7),

演習：Bag-of-Words.ipynb > **1. 少ない単語からなる文書間の類似度**

コサイン類似度計算結果

「は」と「。」しか一致しない
「私 は 本 を 読む 。」と「今日 は 晴天 だ 。」の類似度: 0.365

完全に一致
「私 は 本 を 読む 。」と「私 は 本 を 読む 。」の類似度: **1.0**

1単語だけ違うだけ
「私 は 本 を 読む 。」と「私 本 を 読む 。」の類似度: 0.913

完全に一致しない
「私 は 本 を 読む 。」と「明日 雨 が 降る ！」の類似度: **0.0**

長さが倍で構成要素の比率は変わらない
「私 は 本 を 読む 。」と「私 は 本 を 読む 。本 を 私 は 読む 。」の類似度: **1.0**

演習：Bag-of-Words.ipynb > **1. 少ない単語からなる文書間の類似度**

小説間の類似度

宮沢賢治の「銀河鉄道の夜」「風の又三郎」と太宰治の「人間失格」の3作品について、 お互いがどの程度近いかを文書ベクトルのコサイン類似度で評価してみよう

語彙サイズ: 7721 **→文書ベクトルの次元は7721**

	銀河鉄道の夜	風の又三郎	人間失格
銀河鉄道の夜	1.0	0.9402969	0.87744071
風の又三郎	0.9402969	1.0	0.81866464
人間失格	0.87744071	0.81866464	1.0

- 「銀河鉄道の夜」は「人間失格」よりも「風の又三郎」のほうが似ている
- 「人間失格」は「風の又三郎」より「銀河鉄道の夜」のほうが似ている

演習：Bag-of-Words.ipynb > **2. 小説間の類似度**

トピック分析
tf-idfによる重要語抽出

演習：TopicAnalysis2.ipynb

文書ベクトルの改良

- 先ほどの文書ベクトルは単語の出現回数を数えていた
- その文書にたくさん登場するからと言って、その単語が重要とは限らない！
  - 「の」はどんな文書でも最頻出単語
  - 「あなた」「私」などもどの小説にもたいてい現れる
  - 「カンパネルラ」といえば・・・『銀河鉄道の夜』！

つまり文書において重要な単語とは…

- **その文書によくあらわれる**=これまでとほぼ同じ

  → Tf (Term Frequency：語の出現頻度）
- **その単語が現れる文書が少ないこと！**

  → iDf (Inverse Document Frequeny: 逆文書頻度）

演習：TopicAnalysis2.ipynb > tf-idfによる重要語抽出

tf-idf値

ある単語 $word_i$ のある文書 doc_j におけるtf-idf値を $\mathrm{tf{-}idf}(word_i, doc_j)$とすると、

$$\mathrm{tf{-}idf}(word_i, doc_j) = \mathrm{tf}(word_i, doc_j) \cdot \mathrm{idf}(word_i)$$

Tf ：頻繁に表れる語のほど重要

$$\mathrm{tf}(word_i, doc_j) = \frac{doc_j \text{に} word_i \text{が現れる回数}}{doc_j \text{の総単語数}}$$

IDf：他の文書には現れず、その文書によくあらわれる語ほど重要

$$\mathrm{idf}(word_i) = \log \frac{\text{文書の総数}}{word_i \text{が現れる文書の総数}}$$

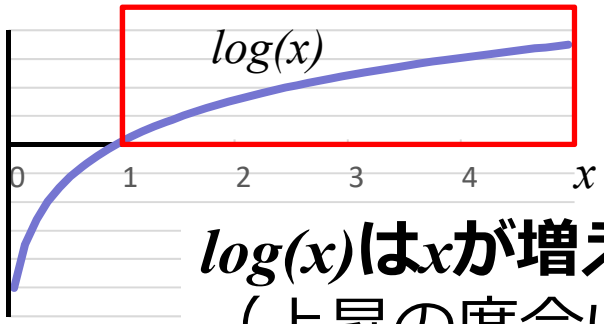
演習：TopicAnalysis2.ipynb > tf-idfによる重要語抽出

iDfをもう少し詳しく

iDf：他の文書には現れず、
その文書によくあらわれる語ほど重要

$$\mathrm{idf}(word_i) = \log \frac{\text{文書の総数}}{word_i\text{が現れる文書の総数}}$$

必ず1より大きい



***log(x)*はxが増えるにつれてなだらかに上昇**
（上昇の度合いは*log*の底によって調整可能）

つまりidfとは…

- その単語がすべての文書に現れるとき0=全く重要じゃない
- 他の文書に現れない単語ほど値が大きくなる

演習：TopicAnalysis2.ipynb > tf-idfによる重要語抽出

演習：TopicAnalysis2.ipynb: tf-idfによる重要語抽出

分析対象：Wikipediaの記事 (wikipedia_wakati.json)

- 6種類のカテゴリ [動物 (animal)、芸術（art)、経済 (economy)、法 (law)、植物 (plant)、政治 (politics)]
- 分かち書きしてjsonファイルにまとめたもの

目的：各カテゴリに属する記事の集合を、それぞれ1つの文書とみなし、各文書（＝カテゴリ）の重要語を抽出

- Tf-iDf値の高い単語を選べばよい
- 機械学習用モジュールscikit-learnのsklearn.feature_extraction.textモジュールのTfidfVectorizerという関数を使って計算

次元の呪い (Curse of dimensionality)

- （数学的）空間の次元が増えるのに対応して問題の算法が指数関数的に大きくなること(Wikipediaより)
  - 例：『フカシギの数え方』は組み合わせ爆発による問題
  - 機械学習においてよいモデルを学習するためには、情報を欠落せずに次元を落とすことが極めて重要！
- テキスト処理で扱うデータはスパースになりがち
  - ほとんど1回しか現れないような単語は分析の役に立たない
  - 書き間違いもある→役に立たない次元が大量発生する！
- 分析に有用な次元だけに絞り込む
  - 出現回数が上位10,000件以下の単語は無視
    → 生成される文書ベクトルは10,000次元となる
  - どの文書にも現れる単語はIDfが低いので無視
    （演習では6カテゴリ中5カテゴリ以下に現れる単語のみ考慮）
  - 特定の文書にしか現れない単語も特殊すぎるので無視
    （演習では6カテゴリ中3カテゴリ以上に現れる単語のみ考慮）

演習：TopicAnalysis2.ipynb > tf-idfによる重要語抽出

検証１：tf-idf値はそのカテゴリにおけるその単語の重要度を表せているか？

ある単語に対する各カテゴリごとのtf-idf値

	「裁判官」	「画家」	「生育」	「細胞」	「融資」	「衆議院」
動物	0.0000	0.0006	0.0057	**0.3792**	0.0000	0.0000
芸術	0.0022	**0.0890**	0.0000	0.0025	0.0030	0.0000
経済	0.0012	0.0000	0.0007	0.0007	**0.0576**	0.0023
法	**0.0750**	0.0013	0.0000	0.0000	0.0000	**0.0213**
植物	0.0005	0.0032	**0.2161**	**0.2527**	0.0000	0.0000
政治	**0.0298**	0.0011	0.0006	0.0000	0.0022	**0.0693**

- 「裁判官」と言えば法か政治（どちらかといえば法）
- 「画家」といえば芸術
- 「細胞」といえば動植物
- 「融資」といえば経済（でも芸術も少しだけ重要らしい…）
- 「衆議院」といえば政治（でも法も重要）

演習：TopicAnalysis2.ipynb > tf-idfによる重要語抽出

検証２：
tf-idf値が高い単語はそのカテゴリを特徴づけているか？

tf-idf値が高い単語上位10件

	1位	2位	3位	4位	5位	6位	7位	8位	9位	10位
動物	細胞	家畜	寄生	個体	飼育	哺乳類	ウシ	化石	昆虫	生殖
芸術	カメラ	スタジオ	漫画	フィルム	美術	デジタル	レンズ	映像	露出	バロック
経済	マルクス	資本	投資	景気	金融	会計	所得	貨幣	銀行	買収
法	監査	憲法	司法	教皇	立法	カトリック	裁判	公法	審査	権力
植物	品種	栽培	種子	細胞	cm	生育	花粉	果実	光合成	乾燥
政治	憲法	主権	監査	権力	議員	独裁	請求	立法	衛星	政党

- その分野に特有の単語が上位に現れている
  - 「品種」「マルクス」「スタジオ」
- 複数カテゴリで上位に挙がっている単語もある
  - 「細胞」は動物・植物
  - 「憲法」は法・政治
- 意外な発見もある？
  - 法に「教皇」「カトリック」

演習：TopicAnalysis2.ipynb > tf-idfによる重要語抽出

タグクラウドによる特定カテゴリの重要語の可視化

カテゴリ「**法**」のタグクラウド

演習：TopicAnalysis2.ipynb > tf-idfによる重要語抽出

タグクラウドによる特定カテゴリの重要語の可視化

カテゴリ「**動物**」のタグクラウド

演習：TopicAnalysis2.ipynb > tf-idfによる重要語抽出

カテゴリ間の類似度

tf-idfベクトルによるカテゴリ間の類似度行列

	動物	芸術	経済	法	植物	政治
動物	1	0.077	0.042	0.026	**0.389**	0.022
芸術	0.077	1	**0.114**	0.076	0.064	0.082
経済	0.042	**0.114**	1	**0.153**	0.045	**0.188**
法	0.026	0.076	0.153	1	0.015	**0.713**
植物	**0.389**	0.064	0.045	0.015	1	0.019
政治	0.022	0.082	0.188	**0.713**	0.019	1

- 「動物」は「植物」と似ている
- 「芸術」はどれともあまり似てないけど、あえて言うなら「経済」
- 「経済」は「芸術」「法」「政治」と同程度似ている
- 「法」と「政治」は似ている

演習：TopicAnalysis2.ipynb > tf-idfによる重要語抽出

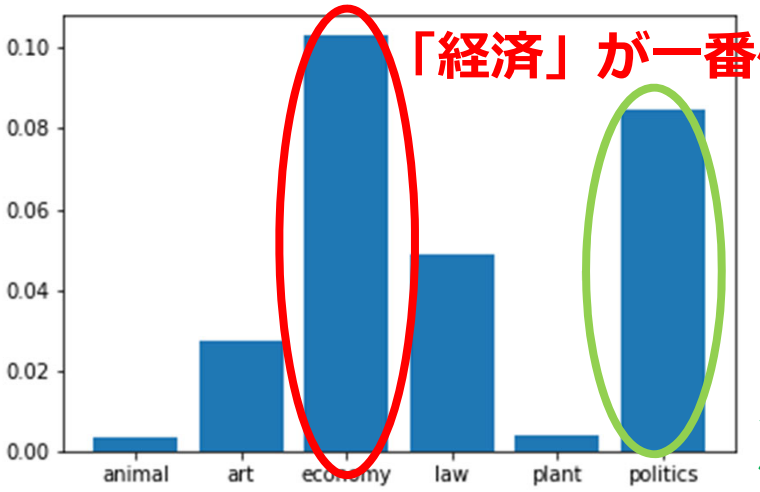
未知の文書はどのカテゴリの記事かを推定しよう

目的：ある未知のWikipediaの記事*sample*がどのカテゴリの記事かを推定したい

Step 1: 学習済みのモデル（vectorizer)を使って***sample***をベクトル化

Step 2: ***sample***のベクトルと各カテゴリから得た文書ベクトルとの類似をコサイン類似度で評価

Wikipedia 「分業 (ID: "204500“)」は経済カテゴリ



「経済」が一番優勢

「経済」と「政治」は元々文書ベクトルが似ているのでこちらも大きく出る

演習：TopicAnalysis2.ipynb > tf-idfによる重要語抽出

LDA を用いた
教師なし学習によるトピック分類

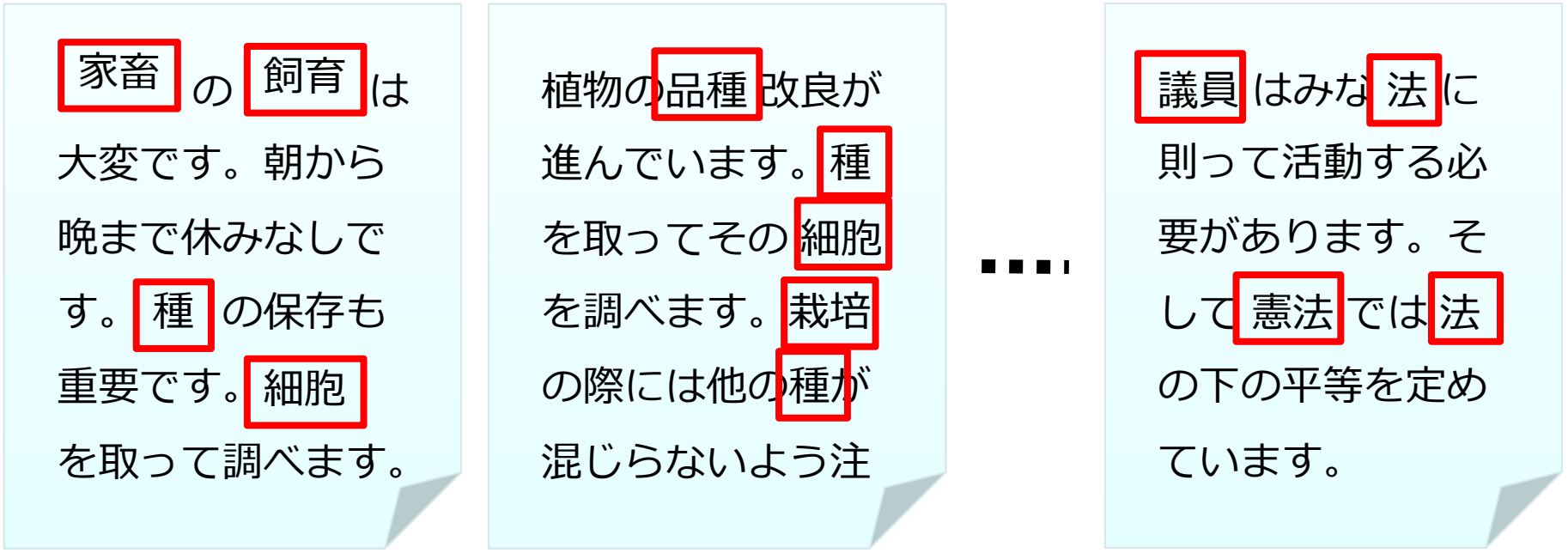
演習：TopicAnalysis3.ipynb

教師なし学習によるトピック分類

- 先のtf-idfの課題ではすでにカテゴリごとに分類された記事が大量に用意されていた
  - 分類済みの文書集合を「教師（supervisor)」として、モデルを教師あり学習（supervised learning)
- 問題：**教師データがない場合**
  - 全く未分類の文書集合が与えられて、「これを適当に分類してね」と言われたら？
  - すでに分類済みであったとしても、「もっと他の分け方ないの？」と言われたら？

  →教師なし学習（unsupervised learning)によるトピック分類
- その一種である**潜在的ディリクレ配分法（LDA: Latent Dirichlet Allocation）**を行う

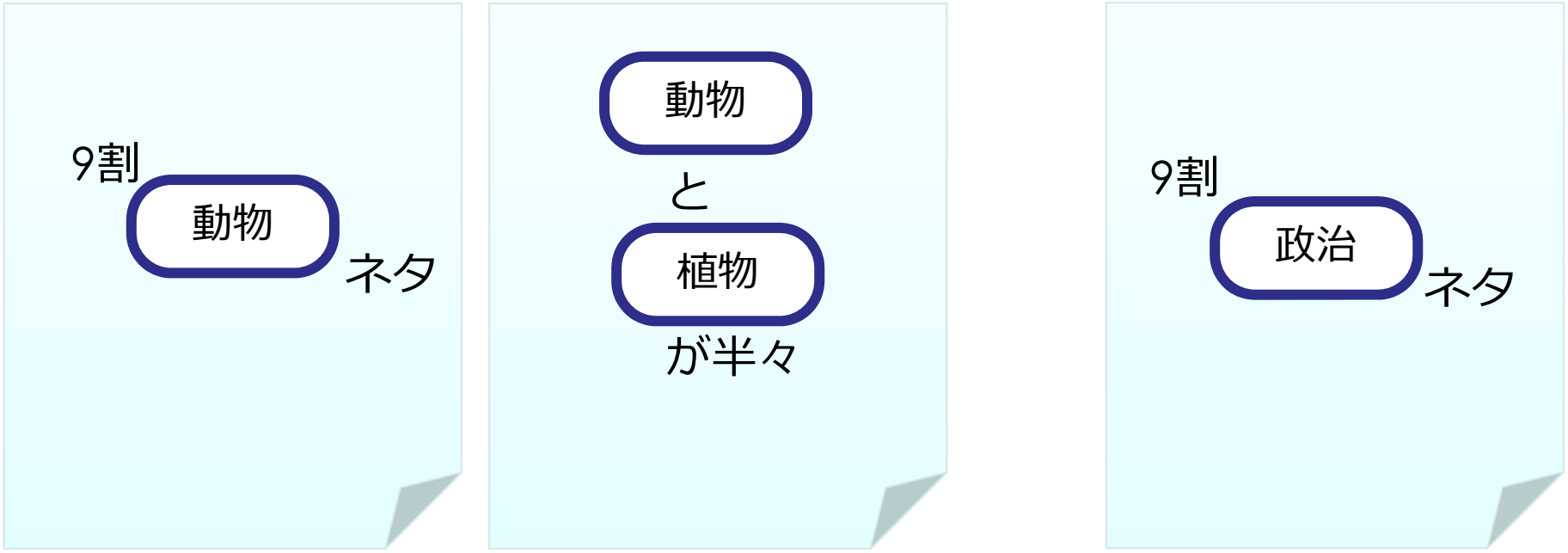
潜在的ディリクレ配分法（LDA: Latent Dirichlet Allocation）



- 文書ごとに単語の出現確率にはバラツキがある（当然）
- このバラツキはなぜ起こるのか？（これがモデル化）
  - 文書は複数のトピックから構成されており
  - トピックごとに単語が出現する確率が違うからだ！

潜在的ディリクレ配分法（LDA: Latent Dirichlet Allocation）

文書を書く時のことを考えると…



潜在的ディリクレ配分法
（LDA: Latent Dirichlet Allocation）

文書を書く時のことを考えると…

文書1

動物

のトピックなら

家畜　細胞

飼育　種

などの単語を
使いがち

文書２

動物　植物

のトピックなら

家畜　細胞　品種

飼育　種　栽培

などの単語を
使いがち

文書M

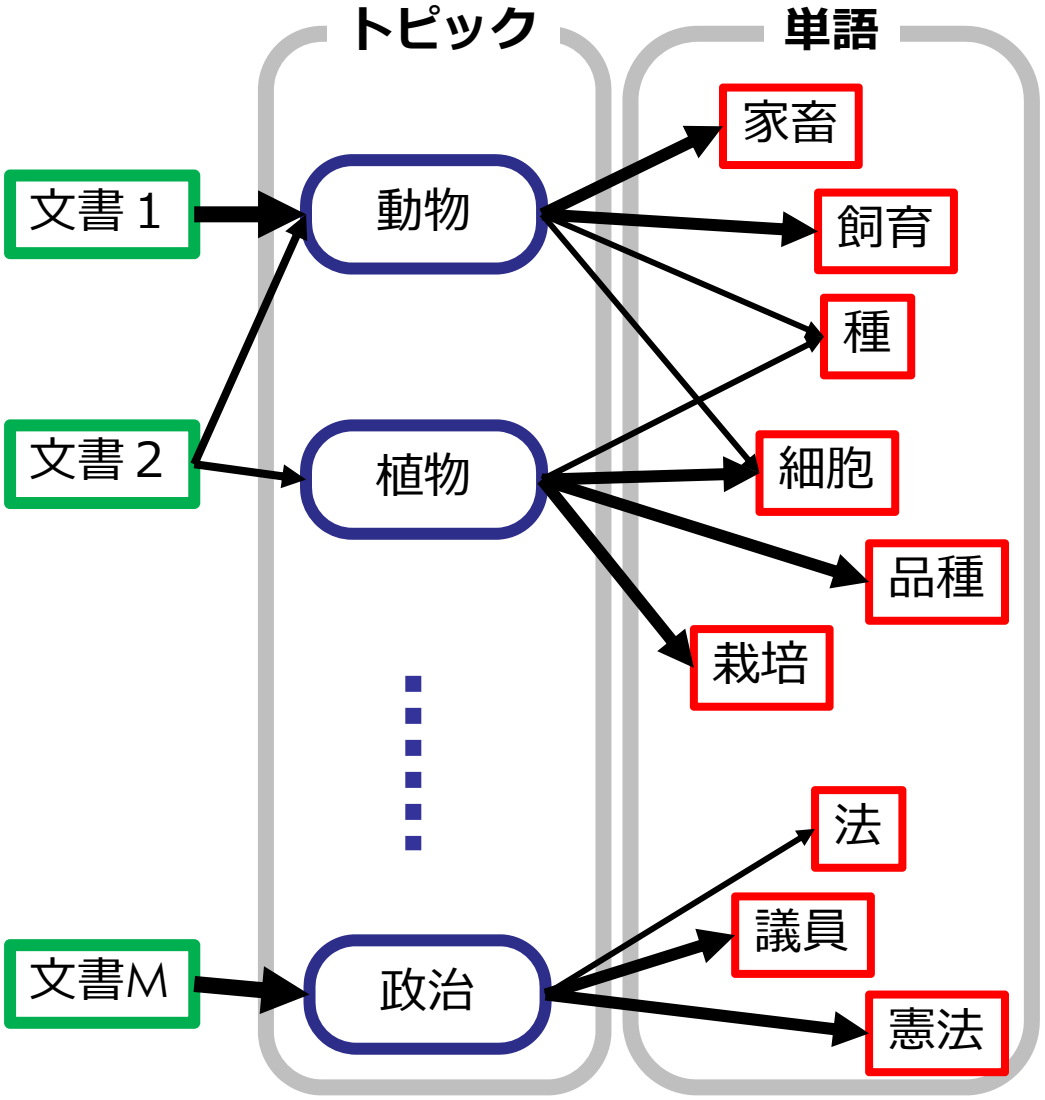
政治

のトピックなら

法　議員　憲法

などの単語を
使いがち

潜在的ディリクレ配分法
（LDA: Latent Dirichlet Allocation）

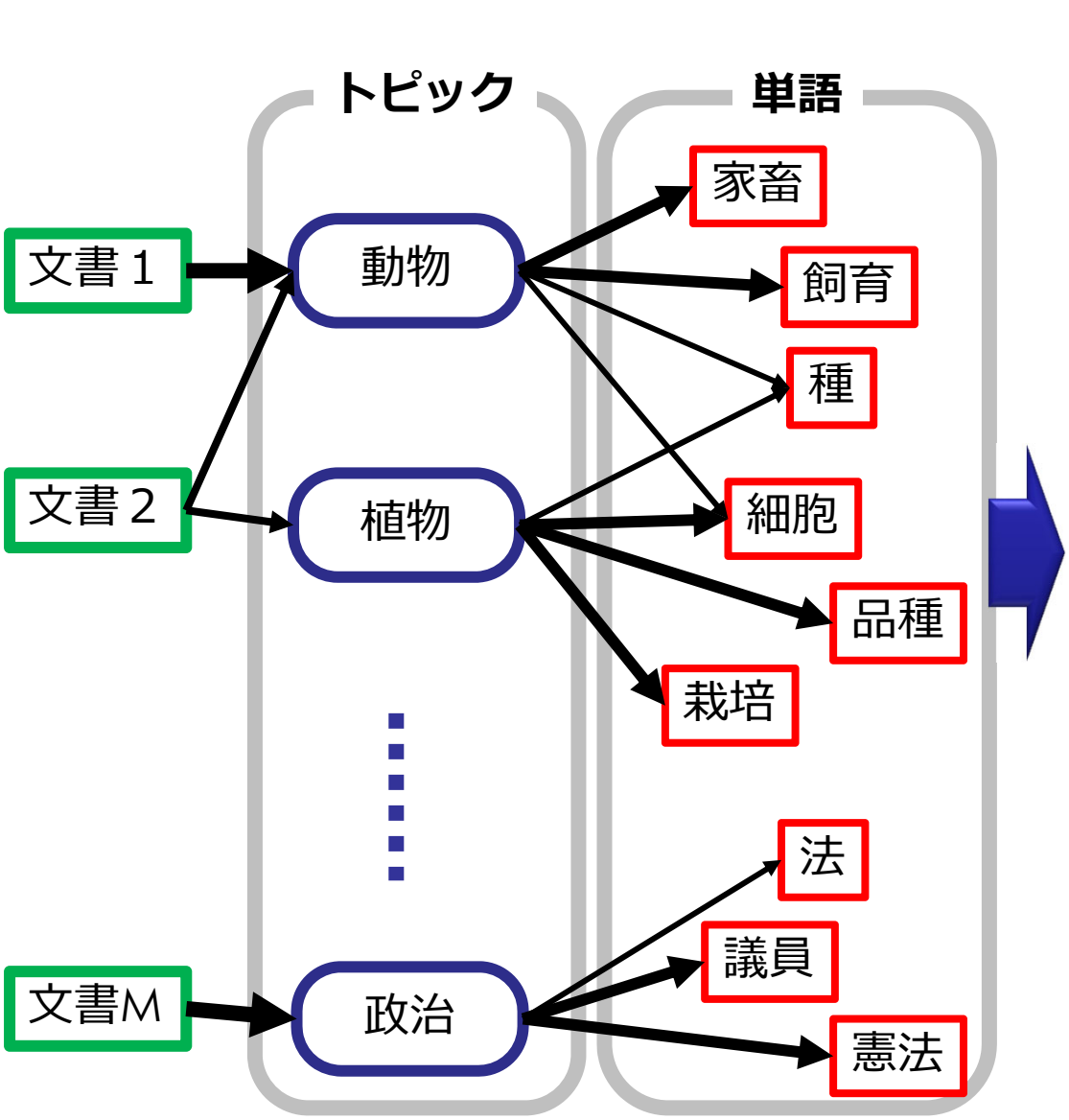


文書は異なる比率による複数のトピックから構成されている

トピックごとに単語が出現する確率が違う

よって各文書の単語の出現確率が異なる

潜在的ディリクレ配分法
（LDA: Latent Dirichlet Allocation）



家畜 の 飼育 は
大変
晩ま
す。
重要
を取

植物の品種改良が進んでいます。種を取ってその細胞を調べます。栽培の

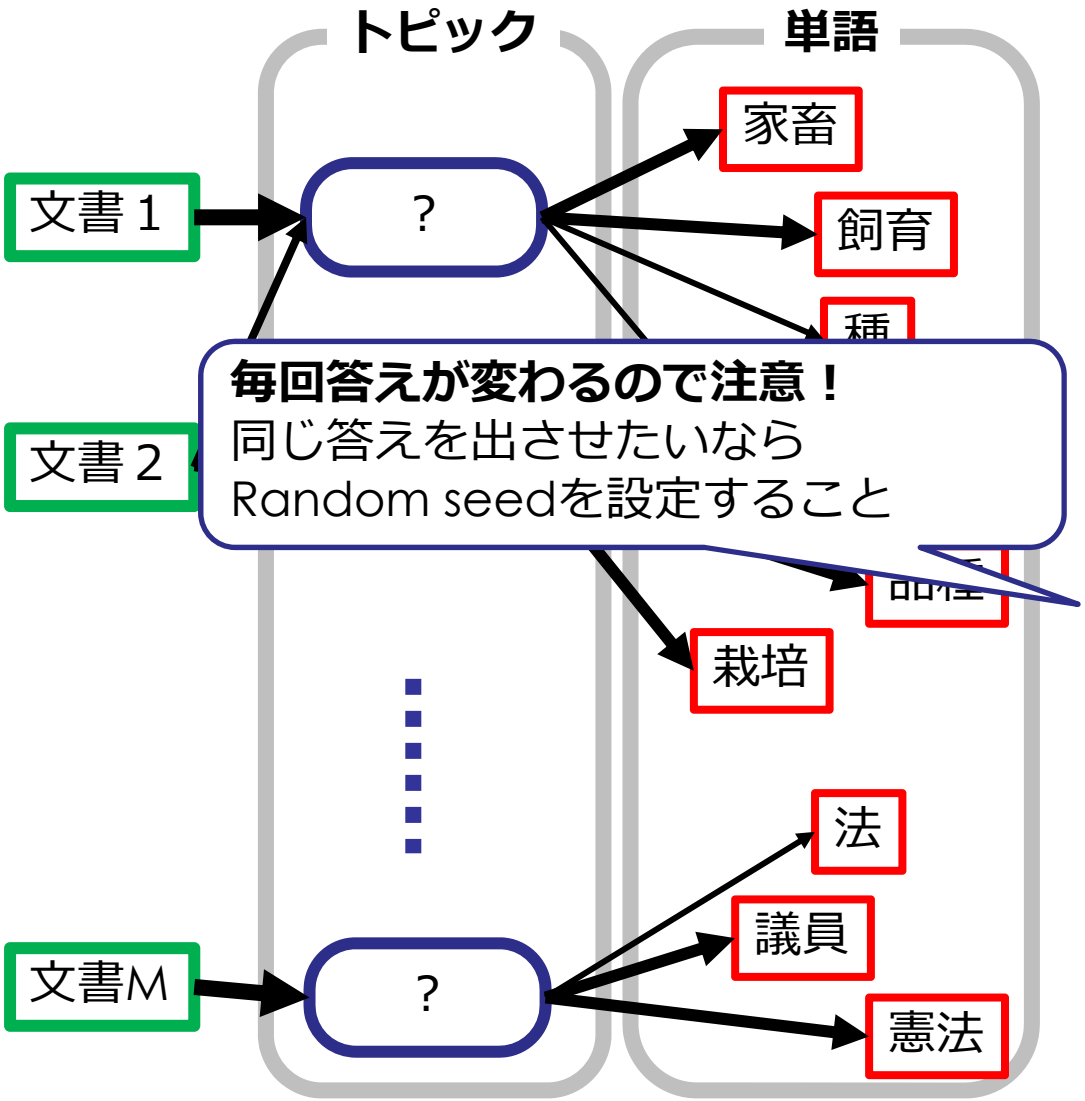
文書1

議員 はみな 法 に則って活動する必要があります。そして 憲法 では 法 の下の平等を定めています。

文書２

文書M

潜在的ディリクレ配分法
（LDA: Latent Dirichlet Allocation）



実際にはトピックは決められていない
→ トピック数を指定する
（決め打ち）

もちろん文書からトピック、トピックから単語の生成確率も与えられていない
→ **初期値としてランダムな確率を与えて**、それを更新していく

どうなるように更新するのか？
・同じ文書の単語は同じトピックから生成されがち
・ある単語を生成する確率は、トピックによって偏りがある

演習：LDAによるWikipedia記事のトピック分類

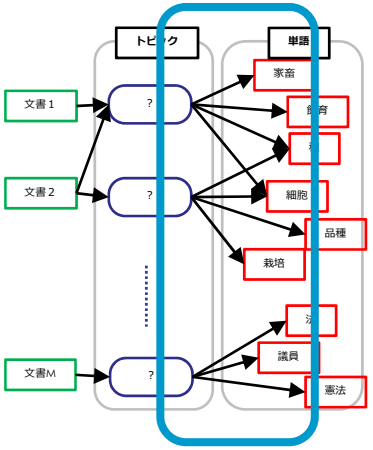
- 2101個のWikipedia記事を分類したい
- カテゴリは知らないものとする（分類性能評価に利用）

前処理：情報を持つ特徴に絞り込んで次元を下げる

- **単語分割**： 日本語では分かち書き、英語ではtokenizationと呼ぶ
- **ストップワード(Stopwords)除去**：あまりに一般的過ぎる単語は、頻出する割にはその文書特有の性質を持たないため除去する
- **ステミング(stemming)**：英語において、語幹のみを取り出す処理
  ex.) running --> run-ning, runs --> run-s, runner --> run-ner
- **見出し語化(lemmatize)**: ran --> runのように、原形に戻す処理
  和文では形態素解析で同様の処理が行える
- **品詞の絞り込み**：特定の機能語に限定（演習では名詞・動詞・形容詞）
  処理前）「脳 科学 （ の う か がく 、 ） と は 、 ヒト を 含む 動物 の 脳 と 、 それ が 生み出す 機能 について 研究 する 学問 分野 で ある 。」
  処理後）「脳 科学 がく ヒト 含む 動物 脳 生み出す 機能 研究 学問 分野 対象」

演習：TopicAnalysis3.ipynb > **LDA を用いた教師なし学習によるトピック分類**

各トピックの重要語とその重み




```
for tpn in range(topic_n):
    print('トピック', tpn, ': ', lda.print_topic(tpn, topn = 5))
```

各トピックについて、どの単語がどの程度重要かを出力
（重要度が高かった上位5単語）

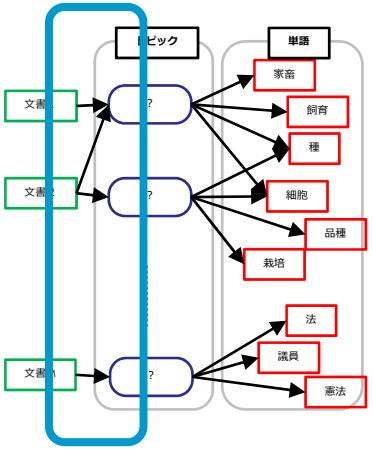
トピック０： 0.019*"-" + 0.014*"主義" + … + 0.011*"植物"
トピック１： 0.057*"項" + 0.042*"委" + … + 0.011*"地方"
トピック２： 0.036*"**憲法**" + 0.022* + … + 0.011*"規定"
トピック３： 0.025*"**経済**" + 0.0 + … + 0.014*"問題"
トピック４： 0.060*"**政治**" + 0.027* + … + 0.015*"主義"
トピック５： 0.040*"**政治**" + 0.021*"国家" + 0.020*"社会" + 0.020*"学" + 0.015*"主義"
トピック６： 0.027*"**文化**" + 0.017*"国際" + 0.012*"適用" + 0.012*"民間" + 0.011*"中国"
トピック７： 0.025*"社会" + 0.018*"女性" + 0.015*"政治" + 0.013*"主義" + 0.011*"史"
トピック８： 0.021*"**花**" + 0.015*"**植物**" + 0.014*"種" + 0.011*"科" + 0.011*"属"
トピック９： 0.025*"**種**" + 0.014*"状" + 0.013*"植物" + 0.011*"葉" + 0.010*"花"

重要語のリストからトピックを解釈

- トピック2は憲法に関する内容
- トピック3は経済に関する内容
- トピック4と5は政治的な内容
- トピック6は文化に関する内容
- トピック8と9は植物に関する内容

演習：TopicAnalysis3.ipynb > **LDA を用いた教師なし学習によるトピック分類**

特定記事に対するトピックの割合



ある文書（「動物学」カテゴリは『動物』）に対するトピックの割合

```
sorted(lda[corpus_bow[4]],
       key=lambda x: x[1], reverse=True)
```

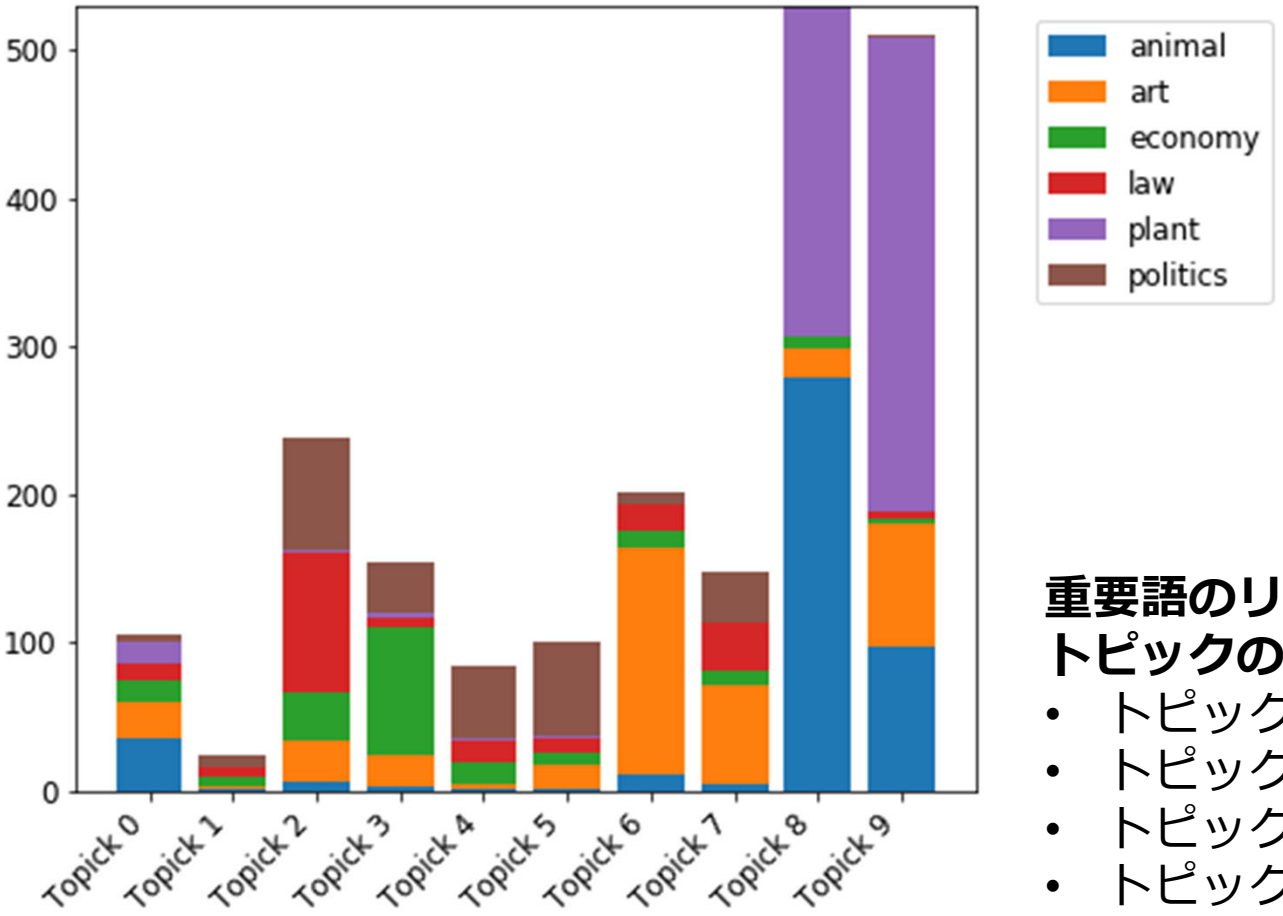
```
[(8, 0.6870042), (5, 0.29615602), (6, 0.01463101)]
```

トピック８が0.68、トピック5が0.29,
トピック６が0.01の重みで混合

重みが最大であったトピックをその記事のトピックとして、
6カテゴリの記事がそれぞれどのトピックに分類されたかを調べてみよう！
同じカテゴリの記事は同じトピックに、
違うカテゴリの記事は違うトピックに分類されていれば成功！

演習：TopicAnalysis3.ipynb > **LDA を用いた教師なし学習によるトピック分類**

各トピックはどのカテゴリの記事を生成したか？

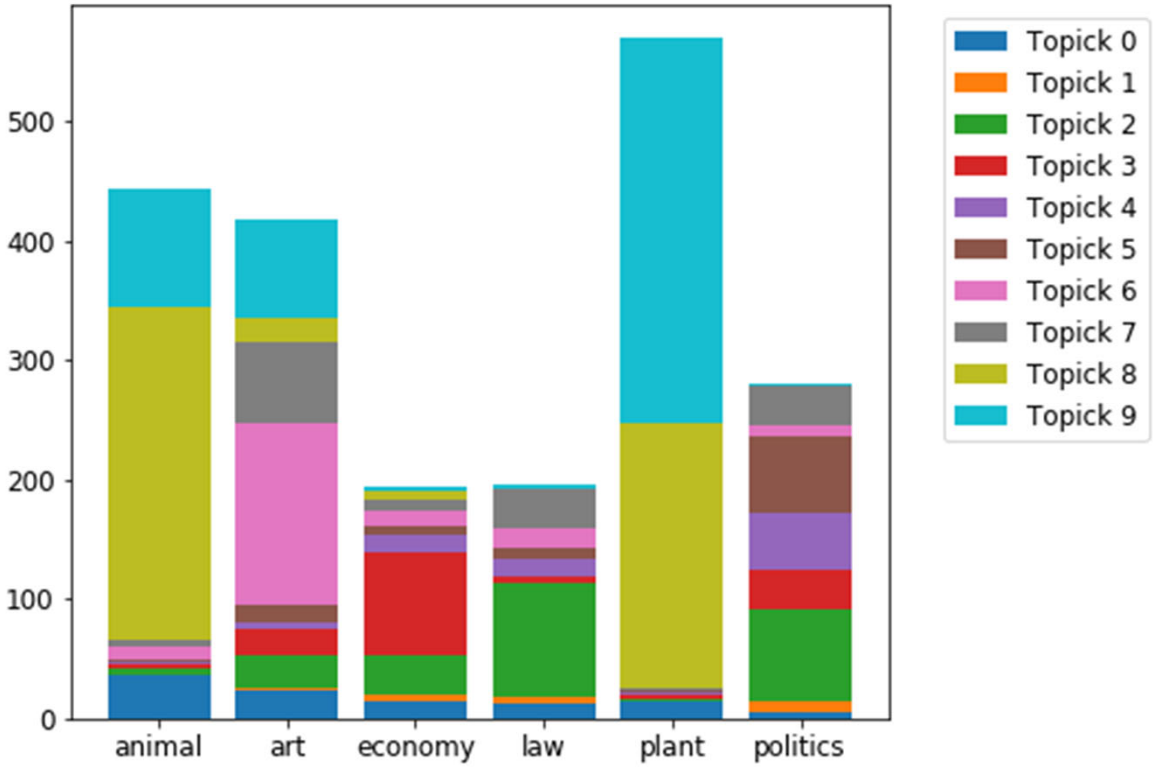


重要語のリストから解釈された
トピックの意味

- トピック2は憲法に関する内容
- トピック3は経済に関する内容
- トピック4と5は政治的な内容
- トピック6は文化に関する内容
- トピック8と9は植物に関する内容

演習：TopicAnalysis3.ipynb > **LDA を用いた教師なし学習によるトピック分類**

同じカテゴリの記事は同じトピックに分類されたか？



重要語のリストから解釈されたトピックの意味

- トピック2は憲法に関する内容
- トピック3は経済に関する内容
- トピック4と5は政治的な内容
- トピック6は文化に関する内容
- トピック8と9は植物に関する内容

演習：TopicAnalysis3.ipynb > **LDA を用いた教師なし学習によるトピック分類**

Word2vecによる Word Embedding

演習：Word2vec.ipynb

Word Embeddingとは？

- 機械学習でテキストを学習させるためには何らかの方法でベクトル化する必要がある
- 1つの方法がone-hotベクトル
  - 大規模なコーパスになると語彙サイズは数万となる
    →単語につき数万次元のベクトルになる
    →次元が高いわりにスパースなので学習が効率が悪い
- 数万次元のone-hotベクトルを数百次元（たとえば200次元）のベクトルに圧縮する方法はないか？

  しかし単に違う数字を割り当てておけばいいわけではない！

  機械学習では数値の大小関係を学習してしまう
  - ある単語Aのベクトルが別の単語Bのベクトルより大きい場合、それは妥当といえるようなベクトル表現でなければならない
  - 似た意味の単語のベクトル同士は近く、意味が大きく違う単語のベクトル同士は遠くあるべき

単語の意味情報を考慮したベクトル化

Word2vec:
その単語の意味はその周辺の単語を見ればわかる

例１）「今日 は 京都 に 出張 で 東京 **[word]** に 行っ た 。」

例２）「出張 の ため 東京 **[word]** から 地下鉄 に 乗っ た 。」

[word]に入る単語は何でしょう？ → 「**駅**」ですよね！

どのような語来るかは、その周辺の語から推定することが可能
→ その語の意味は、その周辺の語によって規定できる

３）「今日 は 京都 に 出張 で 羽田 **[word]** に 行っ た 。」

例４）「出張 の ため 羽田 **[word]** から 地下鉄 に 乗っ た 。」

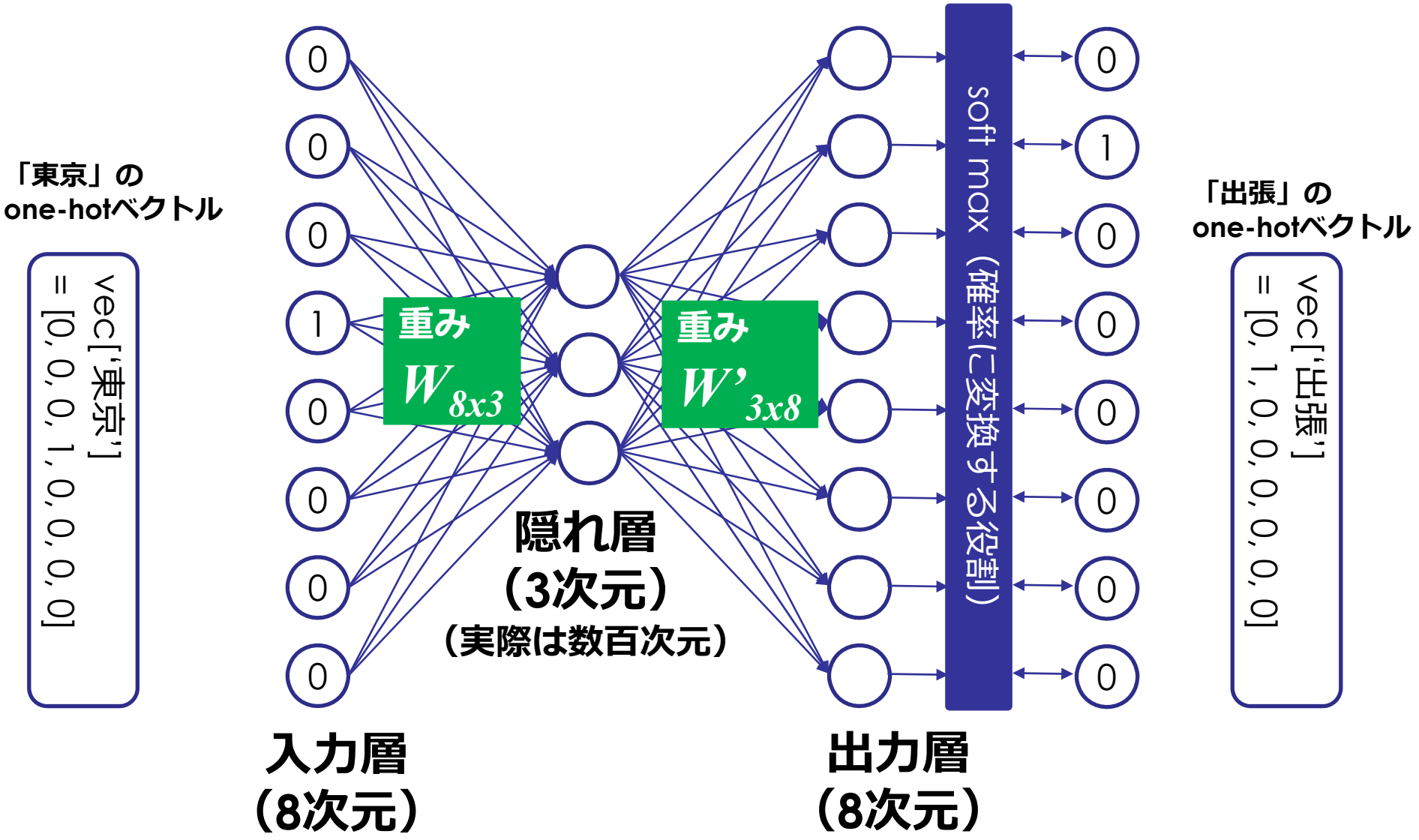
[word]に入る単語は何でしょう？→ 「**空港**」ですよね！

「駅」と「空港」の周辺語はよく似ている
→ 「駅」と「空港」は、「憲法」「家畜」「主義」といった、使いどころの違う単語よりははるかに近い！

- ある $word_i$ 単語が与えられると、 $word_j$ がその周辺語として現れる確率を予測 → **Skip-gram → こちらを解説**
- 周辺の単語リスト $word_{j-N}$ ... $word_{j+N}$ が与えられると、$word_i$ がどの単語であったかを予測 → **CBOW (Continuous Bag-of-Words)**

Word2vec: Skip-gram

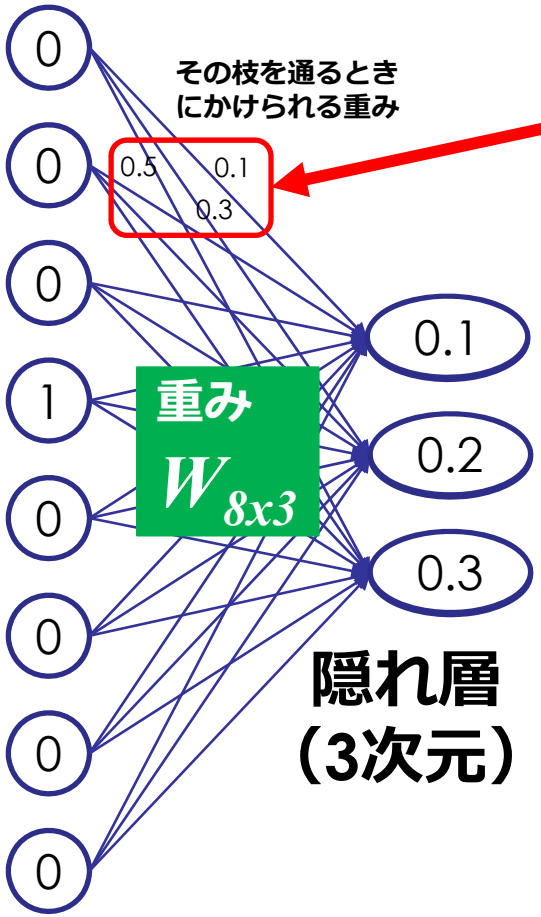
語彙サイズが８だったとする（実際は数百万）



Word2vec: Skip-gram

「東京」の
one-hotベクトル

vec['東京']
= [0, 0, 0, 1, 0, 0, 0, 0]



たとえば
W_{8x3} =
[[0.1 0.3 0.5],
[0.2 0.4 0.5],
[0.5 0.3 0.2],
[0.1 0.2 0.3],
[0.6 0.2 0.3],
[0.4 0.1 0.2],
[0.5 0.5 0.1],
[0.1 0.3 0.2]]なら

$vec['東京'] \cdot W_{8x3}$ =

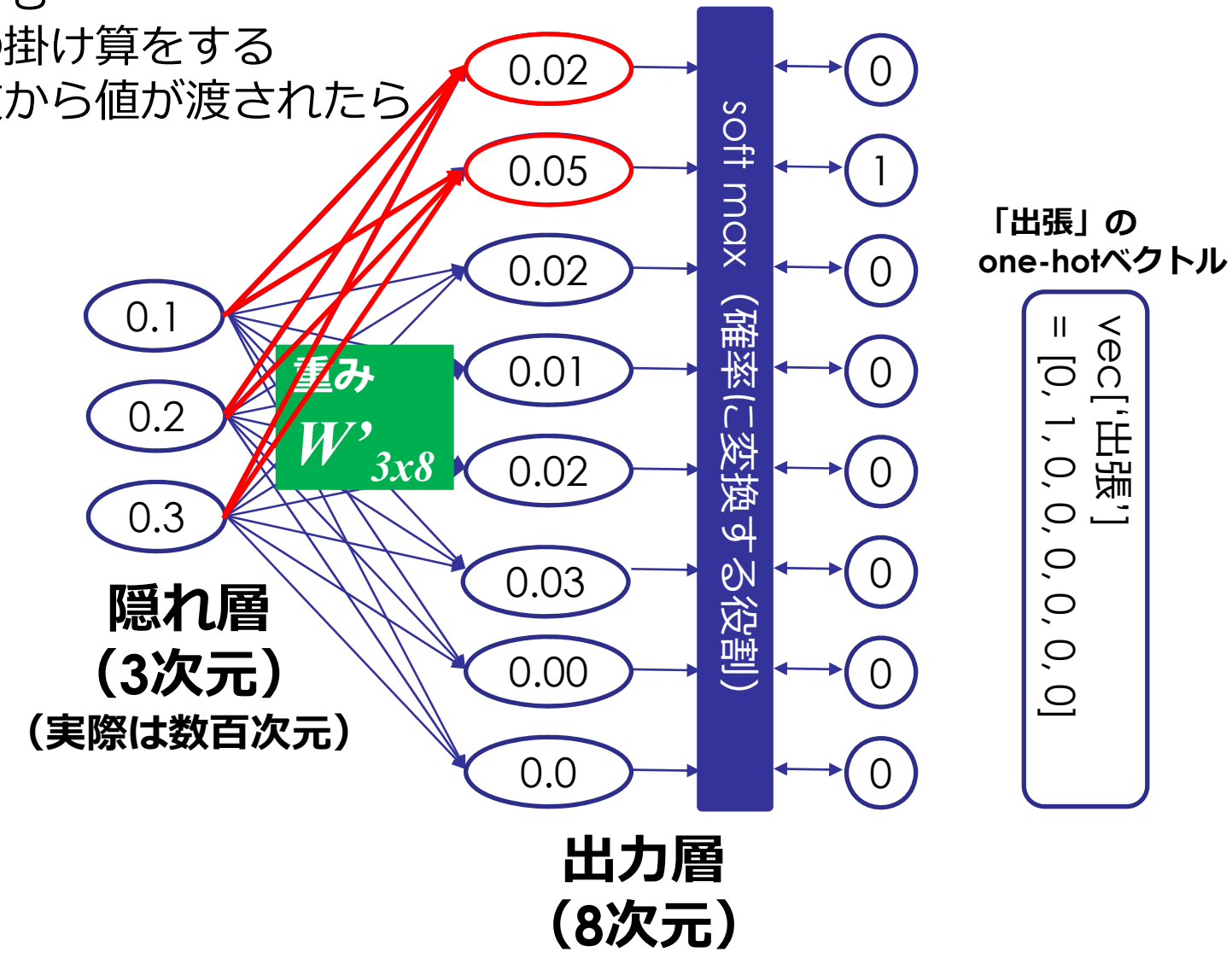
[0 0 0 1 0 0 0 0] ×
[[0.1 0.3 0.5],
[0.2 0.4 0.5],
[0.5 0.3 0.2],
[0.1 0.2 0.3],
[0.6 0.2 0.3],
[0.4 0.1 0.2],
[0.5 0.5 0.1],
[0.1 0.3 0.2]]

=
0×[[0.1 0.3 0.5],
0×[0.2 0.4 0.5],
0×[0.5 0.3 0.2],
1×[0.1 0.2 0.3],
0×[0.6 0.2 0.3],
0×[0.4 0.1 0.2],
0×[0.5 0.5 0.1],
0×[0.1 0.3 0.2]]

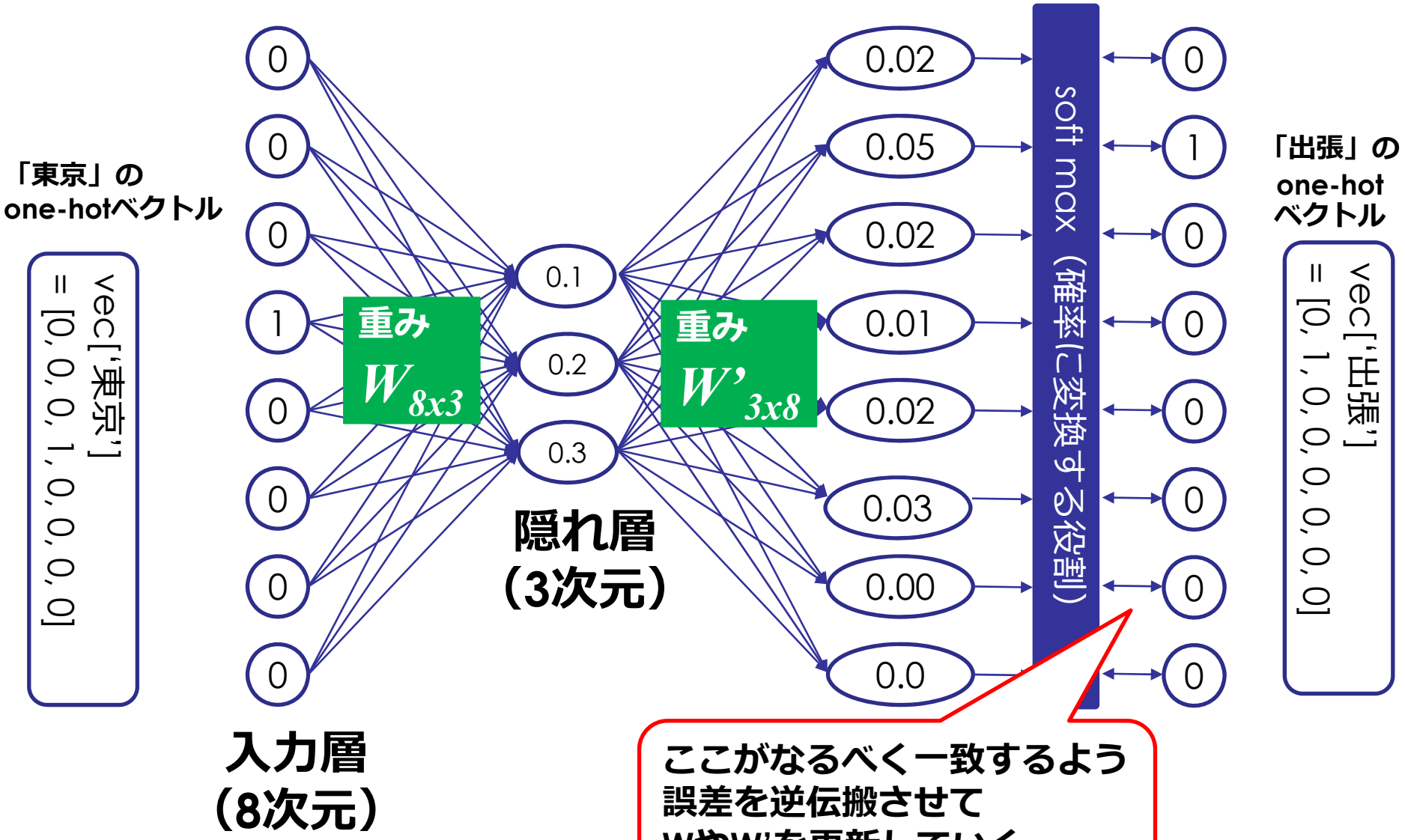
=[0.1 0.2 0.3]

Word2vec: Skip-gram

隠れ層から出力層も
同じように行列の掛け算をする
ただし、複数の枝から値が渡されたら
それらを加算する



Word2vec: Skip-gram



単語ベクトルはどこか？

$vec[\text{‘東京’}] \cdot W_{8x3}$ = 「東京」の one-hotベクトル [0 0 0 1 0 0 0 0] ×

[[0.1 0.3 0.5],
[0.2 0.4 0.5],
[0.5 0.3 0.2],
[0.1 0.2 0.3],
[0.6 0.2 0.3],
[0.4 0.1 0.2],
[0.5 0.5 0.1],
[0.1 0.3 0.2]]

=

0×[[0.1 0.3 0.5],
0×[0.2 0.4 0.5],
0×[0.5 0.3 0.2],
1×[0.1 0.2 0.3],
0×[0.6 0.2 0.3],
0×[0.4 0.1 0.2],
0×[0.5 0.5 0.1],
0×[0.1 0.3 0.2]]

= [0.1 0.2 0.3]

これは「東京」を表す3次元のベクトルである！

つまり…

W_{8x3}

[[0.1 0.3 0.5],
[0.2 0.4 0.5], → 「出張」のベクトル
[0.5 0.3 0.2],
[0.1 0.2 0.3], → 「東京」のベクトル
[0.6 0.2 0.3],
[0.4 0.1 0.2],
[0.5 0.5 0.1],
[0.1 0.3 0.2]]

このそれぞれが、その次元に対応する単語のembedされた3次元のベクトルとなる

演習：Word2vecによるWord Embedding

- トピック分析用ライブラリgensimを用いる
- wikipediaの6つのカテゴリから抽出した、計2101個の記事（分かち書き済み）をコーパスとする
- パラメータ
  - size=200: 200次元のベクトルを計算
  - mini_count=20: コーパスに20回以上現れる単語のみ考慮
  - batch_words=10000: コーパスを10000単語ごとに分けて学習
  - iter=5: 全コーパスを学習するのを5回繰り返す
  - window=15: 前後15単語との関係を考慮する
- 使ってみよう
  - コーパスに含まれており、かつ学習対象となった単語についてしかベクトル化されていないので注意！

3. 単語ベクトルの活用

「動物」の類義語

	類似度
門	0.67
類	0.67
脊椎動物	0.64
爬虫類	0.63
節足動物	0.61
生物	0.60
類縁	0.59
有	0.59
ヌタウナギ	0.59
鳥類	0.58

「日本」の類義語

	類似度
中国	0.66
米国	0.63
沖縄	0.63
欧米	0.62
インドネシア	0.57
アメリカ合衆国	0.57
広く	0.56
琉球	0.56
列島	0.56
諸島	0.55

「カメラ」の類義語

	類似度
フィルム	0.95
デジタル	0.90
TTL	0.87
機種	0.84
撮影	0.84
素子	0.83
画像	0.83
レンズ	0.83
露出	0.83
撮像	0.82

演習：Word2vec.ipynb > **Word2vecによるWord Embedding**

3.2 ベクトルの加算・減算

「国会」-「日本」＋「米国」

('下院', 0.801490545272827 1)
('任期', 0.798736572265625)
('理事', 0.788125455379486 1)
('大臣', 0.787896156311035 2)
('会議', 0.787381410598754 9)

「生育」-「植物」＋「動物」

('飼育', 0.665465950965881 3)
('獣', 0.657667040824890 1)
('居住', 0.625587224960327 1)
('放牧', 0.591455519199371 3)
('ゾウ', 0.587216973304748 5)

「ロンドン」ー「イギリス」＋「日本」

('東京', 0.739454329013824 5)
('日経', 0.685463726520538 3)
('サイト', 0.685226619243621 8)
('毎年', 0.669104337692260 7)
('記念', 0.659624695777893 1)

「ロンドン」ー「イギリス」＋「フランス」

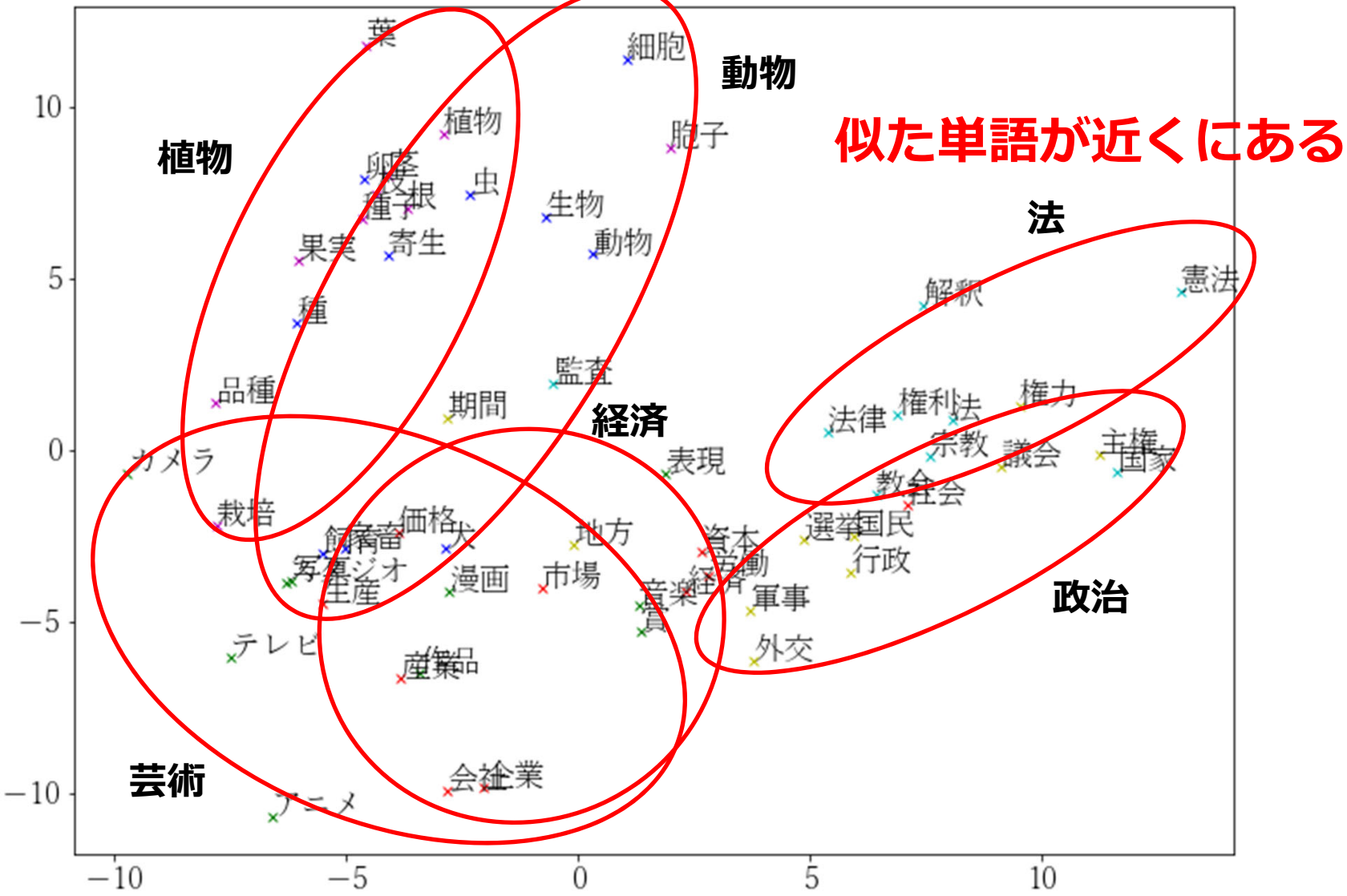
('パリ', 0.861177682876586 9)
('ベルリン', 0.847752094268798 8)
('ベルギー', 0.841508984565734 9)
('結成', 0.820870399475097 7)
('ブリュッセル', 0.812946975231170 7)

Word2vecによる単語ベクトルが、四則演算の成り立つ空間を構成していることを示す

演習：Word2vec.ipynb > **Word2vecによるWord Embedding**

3.3 単語ベクトルの可視化

PCAにより2次元に圧縮



第５回の課題：
宮沢賢治の小説を詳しく分析してみよう

課題１：課題１：*tf-idf*モデルの学習

- 宮沢賢治の全作品を使って、各作品を*1*文書としたときの*tf-idf*を学習
- *sklearn.feature_extraction.text*の*TfidfVectorizer*を利用
- *TfidfVectorizer*のパラメータは以下の通り
  - *max_features=1000 (*語彙サイズは最大*1000*個*)*
  - *max_df=5*　（*5*つ以下の小説までに現れる語彙）
  - *min_df=3*　（*3*つ以上の小説に現れる語彙）

課題２：「銀河鉄道の夜」の重要語の抽出

- 小説「銀河鉄道の夜」の*tf-idf*値が大きい単語上位*10*単語とその*tf-idf*値
- 各行にカンマ区切りで単語と*tf-idf*値が並んだ形式で、*ex5-tfidf.txt*という名前のファイルに出力

発展課題：他の作品について同じことをやってみよう

- **評価には反映しません**
- 「銀河鉄道の夜」とどのような違いがでるか確かめてみてください