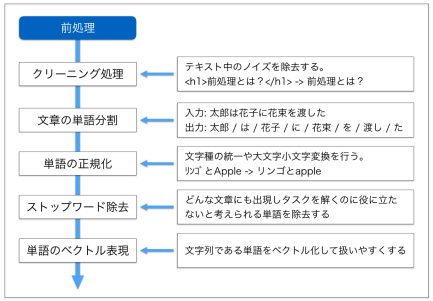
Week13 : Word2Vecについて

- Week10のつづき
- 実習課題のための知識
  - one-hot表現
  - 分散表現
  - word2vec
  - doc2vec

4. ベクトル表現

- テキストデータをベクトル表現にすることで，分析や解析が容易になる.
  → 単語ベクトル
  → 素性（特徴）ベクトル（単語や文章以外にも用いる呼称)
- ベクトル表現にする前にデータを整える必要がある.
  →前処理，データクレンジング， etc.



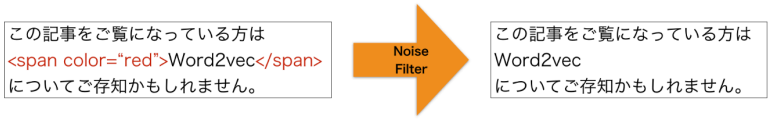
4. ベクトル表現

- 特徴的な単語の情報を数値化（特徴量）した表現

1. **文書を単語ごとに分割**
   クリーニング，単語分割(形態素解析)，正規化，ストップワード除去，表記揺れを整える（ステミング）
2. **特徴となる単語を見つける(単語以外でも構わない．例：属性など)**
   文書で使われる独特の単語，他の文書には出現しにくい単語
   助詞などは除外
   →出現頻度，TF-IDFなどで単語の特徴量を求める.
3. **特徴を数値としてベクトル形式で表す**
   例：文書S1, 文書S2に「桃太郎」「姫」「川」「月」「おばあさん」を素性とする場合，S ={桃太郎, 姫, 川, 月, おばあさん}それぞれの出現頻度をベクトル表現にすると，
   S1={3, 0, 2, 0, 3}，　S2={0, 4, 1, 3, 1}
   S1={1, 0, 1, 0, 1}，　S2={0, 1, 1, 1, 1}　などと表現する.

4. ベクトル表現

- テキストのクリーニング
  - テキストに含まれるノイズを除去
  - JavaScriptのコードやHTML, XMLのタグなど
  - 正規表現を用いる
  - pythonのライブラリを利用する



Pythonにはクリーニングを行うのに便利なライブラリがある.
例： Beautiful Soup, lxml
正規表現をリアルタイムに確認するエディタ（英語のみ）
例：https://regex101.com/

4. ベクトル表現

- 単語の正規化
  - 文字の統一
    - アルファベット→小文字に揃える
    - 半角文字→全角文字に揃える

iPhoneだって半角カナを打てる。(・∀・)ｲｲﾈ!! → Replace → iphoneだって半角カナを打てる。(・∀・)イイネ!!


4. ベクトル表現

- 単語の正規化
  - 数字の置き換え（→ 例えば，全て0へ）~ 数値が重要ではない場合

終値は前日よりも39・03ドル（0・19%）高い2万0689・24ドル。 → Replace → 終値は前日よりも0・0ドル（0・0%）高い0万0・0ドル。


4. ベクトル表現~

- 単語の正規化
  - 辞書を用いた単語の統一
    - 単語を大行的な表現に置き換える.

ソニーの製品が好き！やっぱりSonyだよな！ → Replace → Sonyの製品が好き！やっぱりSonyだよな！

他にも

つづり揺れ	“looooooooooool" → ” lol”
省略語の処理	“4eva” → “forever”
口語表現の代表化	“っす” → “です”
表記揺れ	“はんぶんづつ” → 半分ずつ


4. ベクトル表現

- 単語の分割
  - 形態素解析器を用いる.
  - 原型を用いると語彙数を減らすことができる.

彼女と国立新美術館へ行った。

↓ 形態素解析

彼女	と	国立	新	美術館	へ	行っ	た	。
名詞	助詞	名詞	接頭詞	名詞	助詞	動詞	助動詞	記号

原形にした場合

彼女	と	国立	新	美術館	へ	行く	た	。
名詞	助詞	名詞	接頭詞	名詞	助詞	動詞	助動詞	記号

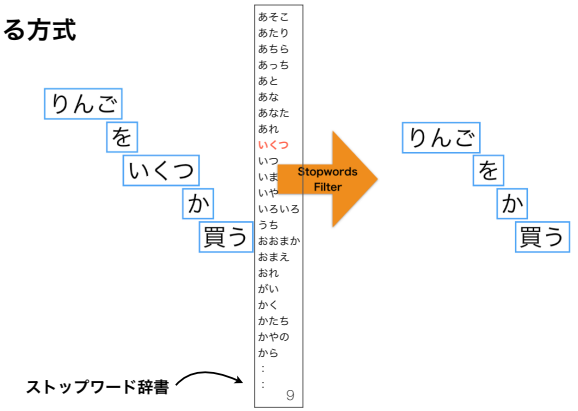
形態素解析器

例： MeCab, Juman++, Janome など


4. ベクトル表現

- ストップワードの除去
  - 処理対象外　（役にたたない語）
    - 助詞や助動詞など　（目的に応じて決めるとよい）
  - 辞書による方式，出現頻度による方式

辞書による方式

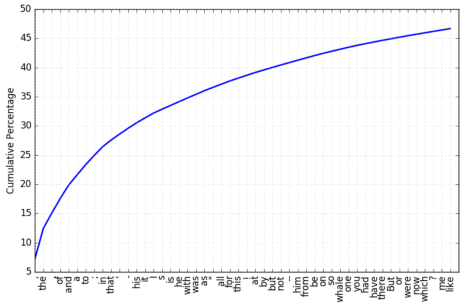


4. ベクトル表現

- ストップワードの除去

出現頻度による方式

- 高頻度（場合によっては低頻度）の単語を除去



英語の本の最も頻出する50単語の累計頻度

4. ベクトル表現

- ベクトルの要素

単語の特徴量を求めベクトルの要素とする.

- 出現頻度
- tf-idf
- 属性（数値化した値）
- 共起率
- 相互情報量

など

処理に**適切な特徴**をベクトル化する→特徴エンジニアリング

1.one-hot表現

文章：すもも　も　もも　も　もも　の　うち

	すもも	も	もも	の	うち
Index	0	1	2	3	4

one-hot表現

Index: 0 1 2 3 4

「すもも」→ [1 0 0 0 0]
「も」→ [0 1 0 0 0]
「もも」→ [0 0 1 0 0]
「も」→ [0 1 0 0 0]
「もも」→ [0 0 1 0 0]
「の」→ [0 0 0 1 0]
「うち」→ [0 0 0 0 1]

2. 分散表現

- 単語間の関連性や類似度に基づくベクトルで単語を表現
- 文脈を考慮

200要素程度

単語i	文脈 j（前後の単語）			
王	0.03	0.58	0.25	…
男	0.38	0.91	0.02	…
ロンドン	0.97	0.09	0.34	…
女	0.22	0.75	0.06	…

共起率

似ている単語の分散表現のベクトルは近い値になる.

- 単語を表すベクトル同士で足し算，引き算が可能

例：「王」- 「男」 + 「女」 = 「女王」

単語数(i)が増えても次元(j)は大きくならなくて済む

2. 分散表現

- 単語間の関連性や類似度に基づくベクトルで単語を表現
- 文脈を考慮

200要素程度

単語i　　文脈 j（前後の単語）

・単語の周辺文脈から、単語の意味を表現するベクトルを獲得する
・Skip-gram や CBOW などのモデル(Word2vec)
・→学習された単語ベクトルには、
  - 「意味が似た単語同士は近い値を持つ」
  - 「ベクトル同士の演算によって意味の関係が表現できる」

例：「王」- 「男」 + 「女」 = 「女王」

単語数(i)が増えても次元(j)は大きくならなくて済む

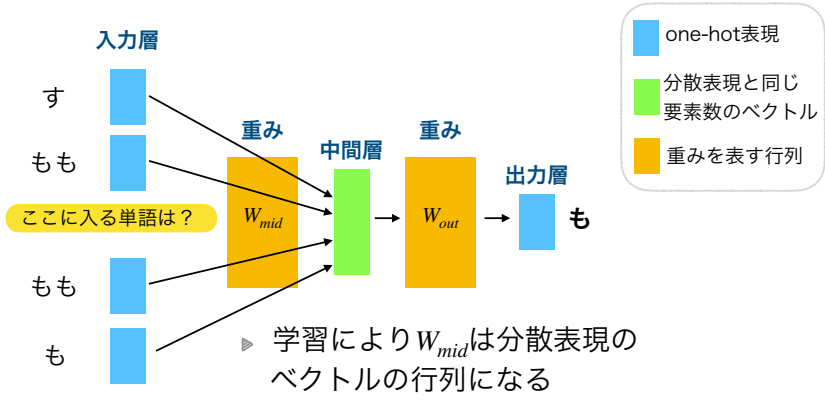
3. word2vec

- Word to Vector
- 分散表現を作成することができるツール
- CBOW(continuous bag-of-words)もしくは，skip-gramというニューラルネットワークが用いられる

- bag-of-words: 出現する単語の集合

3. word2vec ~ CBOW

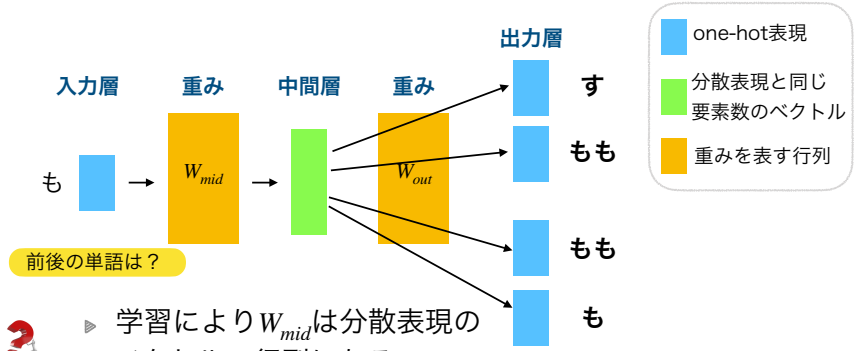
- 前後の単語から対象の単語を予測するニューラルネットワーク
- 学習に要する時間が skip-gramよりも短い



- 学習によりW_{mid}は分散表現のベクトルの行列になる

3. word2vec ~ skip-gram

- ある単語（中央の単語）から前後の単語を予測するニューラルネットワーク
- CBOWよりも学習時間がかかるが，精度がよい



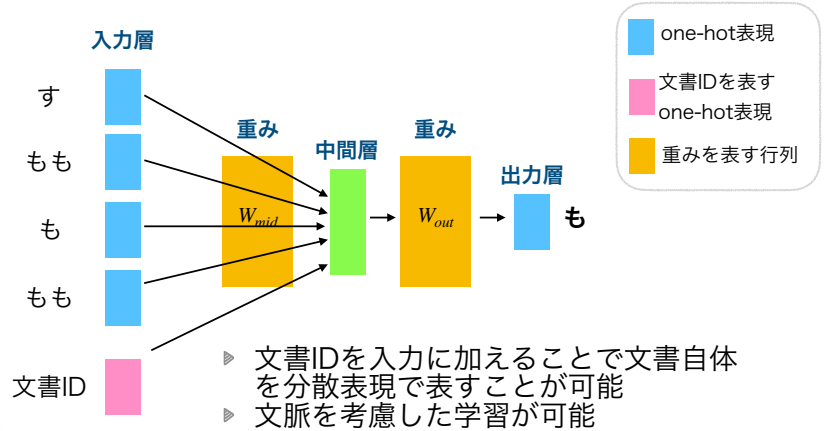
- 学習によりW_{mid}は分散表現のベクトルの行列になる

4. doc2vec

- Document to Vector
- 文や文章の任意の長さのテキストを扱える
- 文や文章に対して分散表現を獲得することができる
- dmpv(distributed memory paragraph vector)もしくは，DBOW(distributed bag-of-words)というニューラルネットワークが用いられる

4. doc2vec ~ dmpv

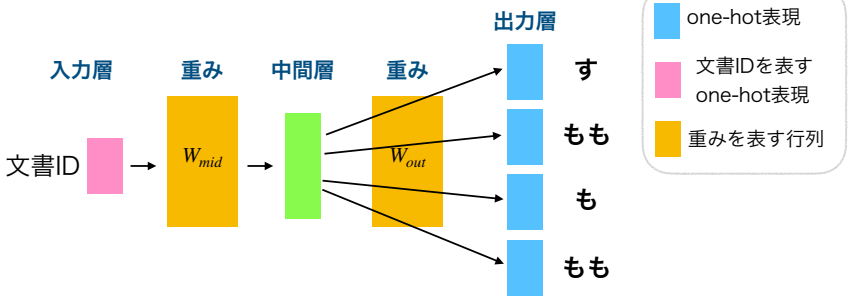
- word2vecのCBOWに似ている



- 文書IDを入力に加えることで文書自体を分散表現で表すことが可能
- 文脈を考慮した学習が可能

4. doc2vec ~ DBOW

- word2vecのskip-gramに似ている
- dmpvより高速



- 該当文書内に含まれる単語を予測できる
- 学習を行うことで，文書自体を分散表現で表すことができる

- word2vecとdoc2vecは gensim (pythonライブラリ)
を用いて実装可能


State-of-the-art (Sota)

キーワード

- **Transformers:**

”Attention Is All You Need”という自然言語処理に関する論文(2017)で提案された新しい深層学習モデル（自然言語処理，画像処理，マルチモーダル，．．．）

- **BERT:** GoogleのJacob Devlinら(2018)

Bidirectional Encoder Representations from Transformers

「Transformerによる双方向に処理を行うエンコーダー」 ベクトル

言語モデルによる事前学習→ e.g. Wikipediaを事前学習

↑をファインチューニング→ e.g. ニュースデータでチューニングし特化できる

- **Hugging Face:**

Transformersを中心としたライブラリ，学習済みの機械学習モデルやデータセットなどを公開