日経記事の解析に文脈ベクトルを使うための環境整備

吉　武　春　光

１．まえがき

筆者が取り組んできた「コンピュータを用いた自然言語の意味処理」において、長い間、使われてきた手法は、語の意味を記述し、語の構文に従って、文の意味を合成し、更に、談話の意味も合成する、というものである。しかし、語の意味を１つ１つ記述している作業が膨大であることから、一般に行われているビッグデータ処理では、意味の合成は行わずに、語の出現頻度を統計処理している。そこで筆者も意味の合成を行わない手法を検討することにし、前稿（吉武 2015）において、日本経済新聞社の新聞記事１８年分（1995年1月～2012年12月）を処理する際の日本語WordNetの可能性を検討した。日本語WordNet は、曖昧さ解消に有効であると判明したが、全ての語の意味を日本語WordNet に記述することには無理があるために、別の手法を検討することにした。

別の手法として、潜在意味解析 Latent Semantic Analysis(LSA）(Landauer et al. 1998)がある。これは、文章の意味を表すために単語の意味を合成していくのではなく、文章と、そこに出現する単語との間には、共通のトピックとなるような意味があると仮定し、その意味を確率的に抽出するという手法である。本研究では、潜在意味解析の中の１つの手法である Word2Vec (Mikolov et al. 2013) が、日本経済新聞社の新聞記事の解析に適用可能かの検証に取り組むものである。

２．Word2Vec による文脈ベクトルの生成

本章では、Word2Vec の概要を述べた上で、Word2Vec を使用するために必要となる、前処理、文脈ベクトル生成処理について述べる。

2.1　潜在意味解析と Word2Vec

潜在意味解析は潜在的意味索引Latent Semantic Indexing(LSI)とも呼ばれている。潜在意味解析は、単語の「意味」は文脈ベクトルで決まるとしており、文脈は文章内頻度のこととしている。例えば、大抵の日本人は、「白鵬が単独首位　琴欧州敗れる」という文を見た時に、相撲という言葉は出ていないにもかかわらず、相撲に関するものだと理解できる。これは、各々の単語は予め潜在的なトピックを持っており、文章中に単語が独立に出現するのではなく、同じトピックを持つ単語が出現しやすい、という考え方である。文脈ベクトルの作り方に幾つかの考え方があり、行列分解系、言語モデル系、ニューラルネットワーク系という方向性がある。

ニューラルネットワーク系の中で、最も注目を集めているのが Word2Vec である。Mikolov (Facebook 所属)は Google 在籍中に Word2Vec を開発し、Google Code のサイトで公開している。Word2Vec の原理は Skip-gram Model である。しかし、Skip-gram Model で使ってある Softmax 関数を、そのまま計算したら膨大な時間がかかるので、Word2Vec では代案として２つの計算方法を実装している。１つが Hierarchical Softmax という計算法であり、もう１つが Negative Sampling という考え方である。

Word2Vec は、C言語で書かれているので、多くのプラットフォームで動かすことができるが、プログラミング言語 Python のライブラリ Gensim への実装もある。C言語版 Word2Vec は文脈ベクトル生成には使いやすいが、文脈ベクトルの参照コマンドがほとんど提供されていないので、研究者が事細かに動かすには、細部まで指定可能な Python Gensim への実装を使うほうが良さそうである。

Word2Vec は、結局、構文や意味の複雑さを、ニューラルネットワークに置き換えたと、捉えることができるようだ。

2.2　Word2Vec を使うために必要な処理の概要

本節では、Word2Vec を用いた文脈ベクトル生成を行うために必要な

・単語切り出し処理

・文脈ベクトル生成処理

について述べる。全体の流れは図１の通りとなる。

2.2.1　MeCab を使った単語切り出し処理

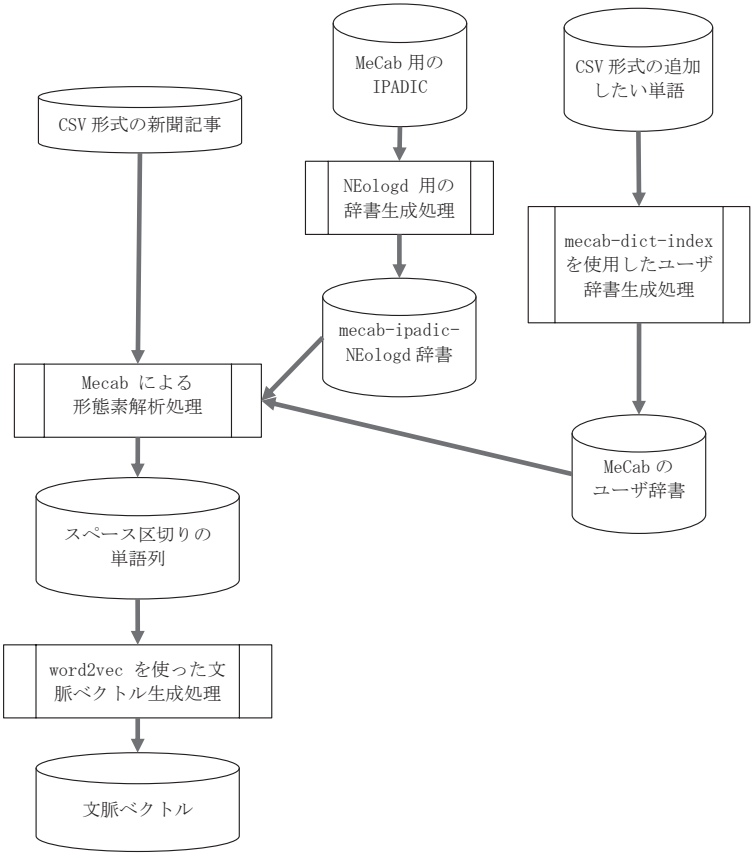

図１　word2vec 文脈ベクトル生成処理の流れ

MeCab に使用する辞書は、今回、新語辞書と、更にユーザ辞書を追加した。

MeCab が標準的に使用する辞書は IPADIC(IPA辞書) というものであるが、2007年までしか更新されていないので、新しい固有表現などの語が登録されていないという問題が生じていた。そこで、Sato(2015) がmecab-ipadic-NEologd : Neologism dictionary for MeCab を公開し始めた。IPADIC に対する差分の形で、毎週２回、更新を行っている。今回は、この mecab-ipadic-NEologd を使用することにした。

辞書への単語追加にはシステム辞書への追加とユーザ辞書への追加という２つの方法がある。しかし、mecab-ipadic-NEologd がシステム辞書を管理しており、ユーザがシステム辞書に追加を行うと混乱する恐れがあると判断したので、本研究では、ユーザ辞書への追加を行うことにした。

具体的には、まず、任意ファイル名.csv ファイルに次の形式で追加したい語を１行に１つという形で記述する。

表層形,左文脈ID,右文脈ID,コスト,品詞,品詞細分類1,品詞細分類2,品詞細分類3,活用形,活用型,原形,読み,発音

ここで、左文脈IDはシステム辞書と同一場所にある left-id.def から該当する ID を選択するが、空にしておくと mecab-dict-index を使って dic ファイルを生成するときに自動的に ID が付与される。右文脈IDとright-id.def も同様である。

さて、MeCab に登録した語の原形が半角スペースを含んでいる場合、例えば下記の「ステップワゴン」の場合の原形「STEP WGN」の場合、後で行う word2vec を使った文脈ベクトル生成処理において、「STEP」と「WGN」が別々の単語として処理されてしまう。そこで、本研究では、原形が半角スペースを含んでいる場合は、それを半角下線に置き直すことにした。つまり、下記の「ステップワゴン」の場合の原形は「STEP_WGN」

とした。

次に、追加した語の一部を図２に示す。

```
フィット,,,3000,名詞,固有名詞,一般,*,*,*,FIT,フィット,フィット
ステップワゴン,,,3000,名詞,固有名詞,一般,*,*,*,STEP_WGN,ステップワゴン,ステップワゴン
ｉＰｏｄ　フォト,1288,1288,4607,名詞,固有名詞,一般,*,*,*,iPod photo,アイポッドフォト,アイポッドフォト
ｉフォン,1288,1288,5072,名詞,固有名詞,一般,*,*,*,iPhone,アイフォン,アイフォン
ｉＰｈｏｎｅ,1288,1288,5072,名詞,固有名詞,一般,*,*,*,iPhone,アイフォン,アイフォン
日本音楽著作権協会,1288,1288,800,名詞,固有名詞,一般,*,*,*,日本音楽著作権協会,ニホンオンガクチョサクケンキョウカイ,ニホンオンガクチョサクケンキョーカイ
```

こうして作った任意ファイル名.csv ファイルに対し次のコマンドを入力し任意ファイル名.dic を生成する。下記では、任意ファイル名を new とした。

```
/usr/local/libexec/mecab/mecab-dict-index -d/usr/local/lib/mecab/dic/ipadic -unew.dic -futf8 -tutf8 new.csv
```

設定ファイル mecabrc において、この new.dic を配置指定する位置を指定した。

```
dicdir = /usr/local/lib/mecab/dic/ipadic
userdic = /home/yositake/new.dic
```

2.2.2　word2vec コマンド

Word2Vec が提供する文脈ベクトル生成コマンドは小文字の word2vec (WORD VECTOR estimation toolkit v 0.1c) である。主要なコマンドオプションは次の通りである。

表１　word2vec コマンドオプション１

オプション名	説明
-train <file>	学習させるデータ。 半角スペースで句切られた単語列。 １行ごとに文脈ベクトルの処理対象となる。
-output <file>	文脈ベクトルの出力先ファイル
-size <int>	文脈ベクトルの次元サイズ (default は 100)
-window <int>	文脈ベクトルを計算する際に、対象語から調べる前後の語の数。default は 5 となっている。大きな値を指定すると、より広範囲の語まで処理対象とする。
-binary <int>	Binary 形式で出力したい場合に 1 と指定する。 default は 0 となっており、テキスト形式で出力される。
-threads <int>	並列計算を行う際のスレッド数。(default は 12)

表２　word2vec コマンドオプション２

オプション名	説明
-hs <int>	Hierarchical Softmax で計算させたい場合に 1 と指定する。default は 0 となっており、下の Negative Sampling で計算する。

-sample <float>	Hierarchical Softmax の際に頻出語をカットするスレッシュホールドを指定する。default は 1e-3 となっており、妥当な値の範囲は 0 から 1e-5 となっている。
-negative <int>	Negative Sampling で用いる例の数。default は 5 となっており、妥当な値の範囲は 3 から 10 となっている。0 を指定すると Negative Sampling を行わない。

表３　word2vec コマンドオプション３

オプション名	説明
-cbow <int>	Bag of words Model を使用するときに 1 と指定する。default は 1 となっており、Skip-gram Model を使用する際は 0 と指定する。

word2vec コマンドを使用する際に Hierarchical Softmax を指定したい場合は必ずコマンドオプション２において「-hs 1」と指定し、Skip-gram Model を指定したい場合は必ずコマンドオプション３において「-cbow 0」と指定する必要がある。オプションで注意が必要なのは、学習させるデータ量と文脈ベクトルの次元サイズの関係である。学習させるデータ量が多い場合は、文脈ベクトルの次元サイズを大きくする方が精度が高くなる。

以下に、word2vec コマンドのパラメータ指定を、状態遷移図として示す。

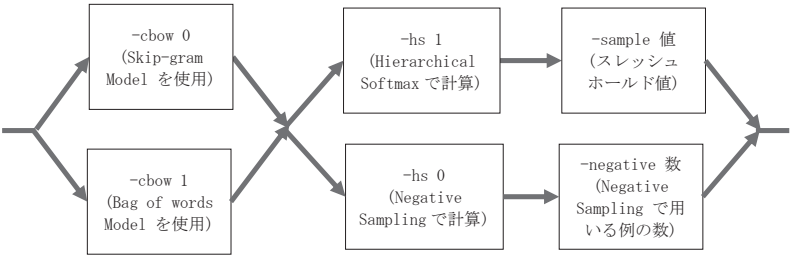

図３　word2vec のパラメータ指定

３．日経新聞記事の前処理と文脈ベクトル生成処理

本研究では、吉武(2015）で用いた日本経済新聞社の本紙（以下、日経データと略す）１８年分（2005年から2012年までで、総容量約５GB）を使用し、Word2Vecを用いた文脈ベクトルの生成を行った。

3.1　日経新聞記事の処理

吉武(2015）で詳述したが、日経データは、月ごとに１つのCSV形式のファイルとなっている。例えば、次が１つの記事である。

"NIRKDB20040429NKM0460","20040429","NKM","日本経済新聞　朝刊","ＰＤ７１４,Ｔ４６８９,＄絵写表記事,＄企業","ヤフー井上社長、「ネット広告・競売成長続く」、価格誤表示防止策急ぐ。","　二〇〇四年三月期決算で最高益を大幅に更新したヤフーの井上雅博社長＝写真＝は日本経済新聞社と会見し、主力事業のネット広告、オークション（競売）事業が今後も大幅な成長を続けるとの見通しを示した。また、二十一―二十二日に同社のショッピングサイトでパソコンの価格が誤って表示された問題を受けて、システム上のチェック体制を強化する考えを明らかにした。
　競売事業について「取引の場としての安全性を向上させれば、事業規模を大きく拡大できる」とみる。サイト上で発行している利用認証コードを郵送に切り替えるなどで「詐欺などのトラブルをゼロに近づける」考え。また「携帯電話やネットに接続できるテレビなどパソコン以外の対応も進める」と話した。
　「ヤフーショッピング」でパソコンの価格が誤って表示された件については、「商品のデータベースの間違いを発見する仕組みがうまく機能していなかった」とし、「サイトに価格を掲載した瞬間におかしいと自動的に判断する仕組みの構築を急ぎたい」と話した。"

図４　日経データのCSV（一部）

カンマ区切りの各欄はダブルクオーテーションで囲まれた値になっており、先頭から次の通りになっている：

```
articleid,date,mediacode,media,bunrui,headline,htmlsource
```

word2vec に学習させるデータは、記事の本文である htmlsource 欄である。但し、
 タグが段落の区切りとして入れてある。

そこで、Python プログラムを使用し、CSV形式記事から、この htmlsource 欄を切り出した後で、MeCab を呼び出すことによってスペース区切りの語の並びを生成した。次に、上記記事を処理した結果を示す。

```
二○○四年三月期決算で最高益を大幅に更新するたヤフーの井上雅博社長＝写真＝は日本経済新聞社と会見する、主力事業のネット広告、オークション（競売）事業が今後も大幅だ成長を続けるとの見通しを示すた。また、二十一―二十二日に同社のショッピングサイトでパソコンの価格が誤るて表示するれるた問題を受けるて、システム上のチェック体制を強化する考えを明らかにするた。　競売事業について「取引の場としての安全性を向上するせるば、事業規模を大きい拡大できる」とみる。サイト上で発行するている利用認証コードを郵送に切り替えるなどで「詐欺などのトラブルをゼロに近づける」考え。また「携帯電話やネットに接続できるテレビなどパソコン以外の対応も進める」と話すた。
「Yahoo!ショッピング」でパソコンの価格が誤るて表示するれるた件については、「商品のデータベースの間違いを発見する仕組みがうまい機能するているないた」とする、「サイトに価格を掲載するた瞬間におかしいと自動的に判断する仕組みの構築を急ぐたい
」と話すた。
```

図５　日経データのMeCab解析結果（一部）

3.2　日経新聞記事のword2vecによる文脈ベクトルの生成処理

Word2vecのsizeパラメータやwindowsパラメータの最適値を見い出すために、次の３つの形式の学習データを用意した。

・記事全体を１入力行とした場合。
・段落区切り記号である
が出現したところまでを１入力行とした場合。
・句点「。」または改行までを１行とした場合。

この３種類のデータに対して、word2vecのパラメータを変えながら、文脈ベクトルの生成処理を行った。

４．Pythonを使ったユーザーインターフェースの作成

本章では、word2vecで生成した文脈ベクトルをアクセスして解析実験を行うために開発したプログラムについて述べる。

4.1　PythonのGensimライブラリ

プログラム言語Pythonはオープンソースの汎用プログラム言語であり、効率が良い読みやすいプログラミングを簡単に書けるようにするという思想に基いており、C、Java、Rubyなどと同様に広く使われている。Pythonの特徴は、核となる機能を必要最小限とし、ライブラリとして色々なものが提供されていることである。また、オブジェクト指向、命令型、手続き型、関数型などの好みのプログラミングスタイルを取ることができる。Pythonライブラリは標準ライブラリ以外に、数多くのライブラリがあるが、トピックモデル用のライブラリGensimの中に、Word2Vec関連のmodels.word2vecがある。

Pythonは２種類のバージョンが存在している。2.7系と3系である。2.7系は長年使用されてきたバージョンであるが、内部構造の見直しが行われ、2008年に3.系に移行した。3.系は2.7系との互換性を捨てた文法体系になっているので、Pythonライブラリで、まだ2.7系までにしか対応していないものが多かったが、2.7系の開発が終了したこともあり、徐々に3系への移行

が進んでいる。Gensim も、公式バージョンは、まだ 2.7系にしか対応していないが、GensimPy3 と称する Python 3系用ライブラリが公開されているので、本研究ではGensimPy3を使用することにした。

4.2　Gensim ライブラリのmodels.word2vec

models.word2vec を使用するには、Python インタープリターの対話画面にて次のコマンドを入力するか、予め書いておいたコマンド群を Python インタープリターに読み込ませるかのどちらかである。models.word2vec の代表的なコマンドは次である：

```
#必要なライブラリをインポート
from gensim.models import word2vec

#変数 dic にセットした文脈ベクトルのファイルから
#変数 model に読み込む。
model = word2vec.Word2Vec.load_word2vec_format(dic, binary=True)

#コサイン類似度を計算する。
#第１引数が positive ワードで、第２引数が negative ワード。
result = model.most_similar(posword, negword, topn=30)
```

図６　models.word2vec の代表的使い方

4.3　文脈ベクトル解析プログラムの作成

さて、実際の解析実験を行う際に、上記のようなコマンドを正確に入力するのは煩雑である。更に、Python インタープリターの対話画面ではなく、Web 画面から文脈ベクトルの選択と検索語の入力を行えることが望まれる。そこで、Python フレームワークを採用し、ユーザインタフェースを作成することにした。

PythonのWebフレームワークには何種類かあるが、本研究では、軽量で、

インストールと使用が簡単なFlask (Version 0.10.1)を使用することにした。

Flask は、基本の Python プログラム、Web の見栄えを担当する template 用のHTMLファイルから、構成される。全体の処理の流れを図７に示す。

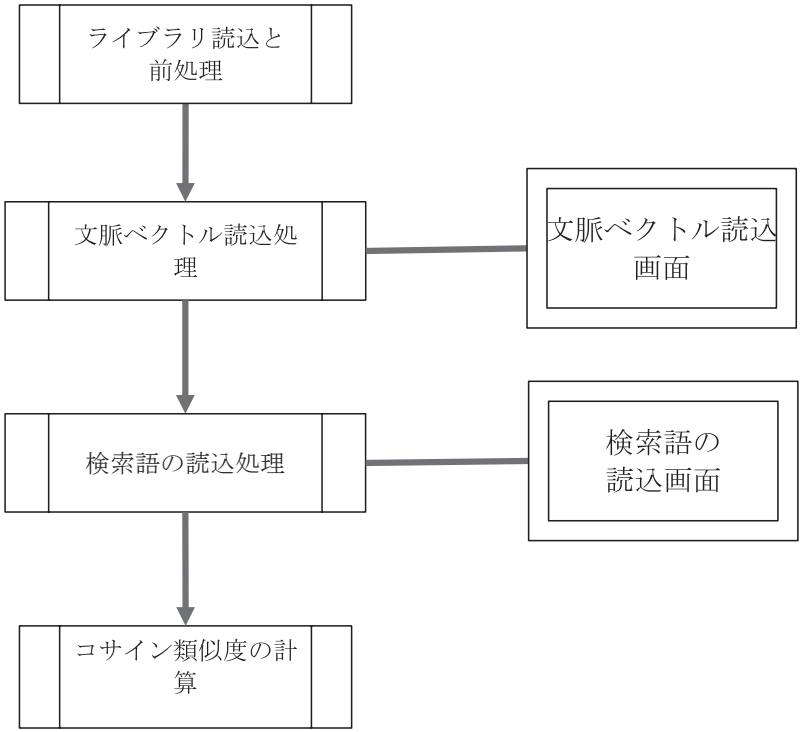

図７　文脈ベクトル解析プログラムの処理の流れ

ユーザーインターフェース画面を図８と図９に示す。

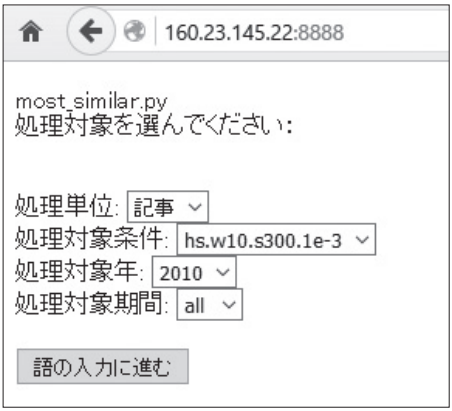

図８　文脈ベクトルデータの読込画面

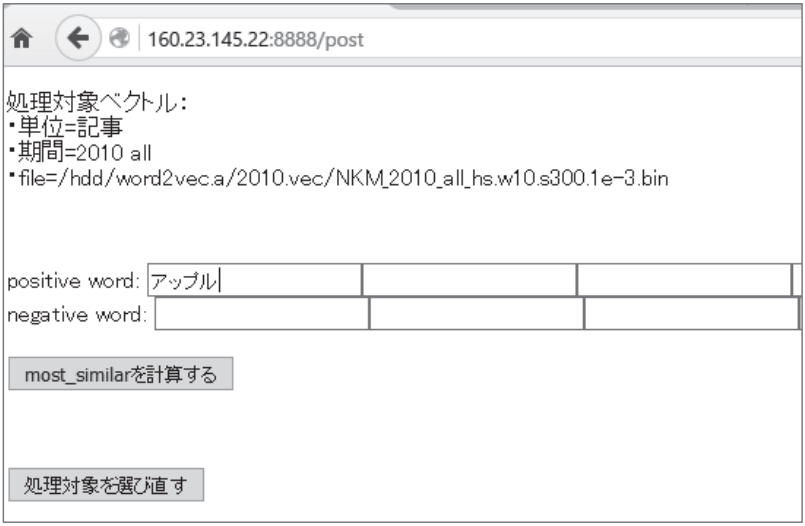

図９　検索語の読込画面

そして、実際の検索結果を図１０に示す。

図10　検索結果の画面

５．検討

上の図１０を見ると、入力された Positive word である「アップル」に対して、iPad や iPhone などが抽出できているのが分かる。実際に解析実験をしてみると、word2vec コマンドのパラメータにより、結果が異なることが判った。例えば、2004年の「iPod」で比較すると、Negative Sampling の方には「ニンテンドーDS」や「プレイステーション・ポータブル」が上位に出現しているが、Hierarchical Softmax の方には、これらは出てこなかった。今回の日経新聞記事の場合には、Negative Sampling を使用した場合は広範囲を捉えているが、Hierarchical Softmax の方は、狭い領域を深く(細かく)捉えていると思われた。

また、-size 値や -window 値や Hierarchical Softmax の -sample 値 などにより結果も変った。定性的ではあるが、今回の日経新聞記事の場合には次の組み合わせが良い結果を出しているように思えた。

・Skip-gram Model 使用
・Hierarchical Softmax 使用
・-size 300
・-window 10

６．あとがき

本研究では、Word2Vec を を使った意味処理の有効性を、日経新聞記事に対して行ったものである。今後は、Word2Vec を発展させた Sentence2Vec や Paraphrase2Vec などに取り組みたい。

なお，本研究で使用した日経データは、本学商学部の 2013 年度の共通図書費を使って購入したものである。購入を認めてくださった商学部の先生方と図書館に感謝します。更に、本学大学院オーバードクターの野間利博氏との議論が本研究を進める原動力になった。野間利博氏が本処理プログラムを使って行う経営学的分析の結果に期待したい。

参考論文

工藤 拓(2002) MeCab: Yet Another Part-of-Speech and Morphological Analyzer
http://mecab.googlecode.com/svn/trunk/mecab/doc/index.html

Flask http://flask.pocoo.org/

Genism https://radimrehurek.com/gensim/

GensimのWord2Vecサイト
http://radimrehurek.com/gensim/models/word2vec.html

Google Codeの Word2Vecサイト
https://code.google.com/p/word2vec/

T.K.Landauer, P.W.Foltz and D.Laham："lntroduction to Latent Semantic Analysis", Discourse Processes, 25, pp.259-284(1998)

MeCab:単語の追加方法
https://mecab.googlecode.com/svn/trunk/mecab/doc/dic.html

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient "Estimation of Word Representations in Vector Space." In Proceedings of Workshop at ICLR, 2013

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. "Distributed Representations of Words and Phrases and their Compositionality." In Proceedings of NIPS, 2013

models.word2vec - Deep learning with word2vec

https://radimrehurek.com/gensim/models/word2vec.html

Python https://www.python.org/

Toshinori, Sato(2015) " Neologism dictionary based on the language resources on the Web for Mecab" https://github.com/neologd/mecab-ipadic-neologd

付録

作成したPythonフレームワークFlaskのプログラム。

```
#!/usr/bin/env python3
import MeCab
import csv
import sys
argvs = sys.argv
t = MeCab.Tagger("")

with open(argvs[1]) as csvfile:
  dataReader = csv.reader(csvfile, delimiter=',')
  for row in dataReader:
    if row[0] != "articleid":
      result = t.parse(row[6])
      for col in result.split('\n'):
        if col != "EOS" and col != "":
          parts = col.split(',')
          print(parts[6], end=' ')
      print()
```

```
#!/bin/sh
echo $1
for name in NKM_$1*; do
echo $name;
/usr/local/research/bin/kiji2mecab.py $name
> /hdd/word2vec.2016.a/$1/`echo $name | sed -e 's/\.csv/\.txt/'`;
done
```

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-

#Flaskなどの必要なライブラリをインポートする
from flask import Flask, render_template, request, redirect, url_for
import numpy as np
from gensim.models import word2vec

target = ''
window = ''
year = ''
term = ''
dic = ''
model = ''

#自身の名称をappという名前でインスタンス化する
app = Flask(__name__)

#ここからウェブアプリケーション用のルーティングを記述
#indexにアクセスしたときの処理
@app.route('/')
def index():
    #read_bin.htmlをレンダリングする
    return render_template('read_bin.html')

#/postにアクセスしたときの処理
@app.route('/post', methods=['GET', 'POST'])
def post():
    title = "処理対象を選んでください"
    if request.method == 'POST':
        #リクエストフォームから「」を取得して
        global target
        global window
        global year
        global term
        global dic
        target = request.form['target']
        window = request.form['window']
        year = request.form['year']
        term = request.form['term']
        if target == '文':
            dic = "/hdd/word2vec.s/" + year + ".vec/NKM_" + year + "_" + term + "_" + window + ".bin"
        if target == '段落':
            dic = "/hdd/word2vec.br/" + year + ".vec/NKM_" + year + "_" + term + "_" + window + ".bin"
        if target == '記事':
            dic = "/hdd/word2vec.a/" + year + ".vec/NKM_" + year + "_" + term + "_" + window + ".bin"
        global model
        model = word2vec.Word2Vec.load_word2vec_format(dic, binary=True)
        #read_word.htmlをレンダリングする
            return render_template('read_word.html', model=model, target=target, window=window, year=year,
term=term
, dic=dic)
    else:
        #エラーなどでリダイレクト
        return redirect(url_for('index'))
```

```
#/wordselにアクセスしたときの処理
@app.route('/wordsel',methods=['GET','POST'])
defwordsel():
    title="語を入れてください："
    ifrequest.method=='POST':
     #リクエストフォームから「」を取得して
     posword=[]
     posword1=request.form['posword1']
     ifposword1!=''andposword1!=None:
        posword.append(posword1)
     posword2=request.form['posword2']
     ifposword2!=''andposword2!=None:
        posword.append(posword2)
     posword3=request.form['posword3']
     ifposword3!=''andposword3!=None:
        posword.append(posword3)
     posword4=request.form['posword4']
     ifposword4!=''andposword4!=None:
        posword.append(posword4)
     posword5=request.form['posword5']
     ifposword5!=''andposword5!=None:
        posword.append(posword5)
     posword6=request.form['posword6']
     ifposword6!=''andposword6!=None:
        posword.append(posword6)
     posword7=request.form['posword7']
     ifposword7!=''andposword7!=None:
        posword.append(posword7)
     #
     negword=[]
     negword1=request.form['negword1']
     ifnegword1!=''andnegword1!=None:
        negword.append(negword1)
     negword2=request.form['negword2']
     ifnegword2!=''andnegword2!=None:
        negword.append(negword2)
     #
     globalyear
     globalterm
     globalmodel
     result=model.most_similar(posword,negword,topn=30)
     #read_word.htmlをレンダリングする
          return render_template('read_word.html', target=target, window=window, year=year, term=term,
dic=dic,result=result,posword=posword,negword=negword)
    else:
     #エラーなどでリダイレクト
     returnredirect(url_for('index'))

if__name__=='__main__':
    app.debug=True#デバッグモード有効化
    app.run(host='160.23.145.22',port=8888)
    #app.run(host='160.23.145.22',port=7777)
    #app.run(host='localhost',port=7777)
```

```
{% extends "layout.html" %}
{% block content %}
 <!-- Form
   ================================================== -->
 <div class="form">
  <div class="container">
   <div class="row">
    <div class="col-md-12">
     <p class="lead">
      most_similar.py Ver.<font color=red>5</font><br />
      処理対象を選んでください：
     </p>
     <form action="/post" method="post" class="form-inline">
      </select> <br />
      処理単位: <select name="target">
        <option>記事</option>
        <option>段落</option>
        <option>文</option>
      </select> <br />
      処理対象条件: <select name="window">
        <option>hs.w10.s300.1e-3</option>
        <option>hs.w10.s400.1e-3</option>
        <option>ns5.w10.s300</option>
        <option>ns5.w10.s400</option>
      </select> <br />
      処理対象年: <select name="year">
        <option>1995</option>
        <option>1996</option>
        省略
        <option>2011</option>
        <option>2012</option>
      </select> <br />
      処理対象期間: <select name="term">
        <option>all</option>
        <option>Q1</option>
        <option>Q2</option>
        <option>Q3</option>
        <option>Q4</option>
        <option>01</option>
        <option>02</option>
        省略
        <option>11</option>
        <option>12</option>
      </select> <br /> <br />
      <button type="submit" class="btn btn-default">語の入力に進む</button>
     </form>
    </div>
   </div>
  </div>
 </div>
{% endblock %}
```

```
{% extends "layout.html" %}
{% block content %}
 <!-- Form
   ================================================== -->
 <div class="form">
  <div class="container">
   <div class="row">
    <div class="col-md-12">
     <p class="lead">
      処理対象ベクトル：<br/>
      ・単位={{ target }}<br/>
      ・期間={{ year }}{{ term }}<br/>
      ・file={{ dic }}<br/><br/>
     {% if posword %}
     positive_word= {{ posword }}<br/>
     {% endif %}
     {% if negword %}
     negative_word= {{ negword }}<br/>
     {% endif %}
     <br/>
     {% if result %}
     similarity=<br/>
     {% for one in result %}
     {{ one }}<br/>
     {% endfor %}
     {% endif %}
    </p>
    <form action="/wordsel" method="post" class="form-inline">
          positive word: <input name="posword1" value="" type="text" /><input name="posword2" value=""
type="text" /><input name="posword3" value="" type="text" /><input name="posword4" value=""
type="text" /><input name="posword5" value="" type="text" /><input name="posword6" value=""
type="text"/><input name="posword7" value="" type="text"/><br/>
          negative word: <input name="negword1" value="" type="text" /><input name="negword2" value=""
type="text" /><input name="negword3" value="" type="text" /><input name="negword4" value=""
type="text"/><input name="negword5" value="" type="text"/><br/><br/>
     <button type="submit" class="btn btn-default">most_similarを計算する
</button>
    </form>
    <br/>
    <br/>
    <br/>
    <form action="/" method="get" class="form-inline">
     <button type="submit" class="btn btn-default">処理対象を選び直す</bu
tton>
    </form>


   </div>
  </div>
 </div>
</div>
{% endblock %}
```