Collective Intelligence

Exercise (on June 24th, 2025)

Word2Vec

Read this document and follow the execution in examples.

(No submission needed)

The former half of this document is based on:
https://towardsdatascience.com/understanding-word2vec-embedding-in-practice-3e9b8985953

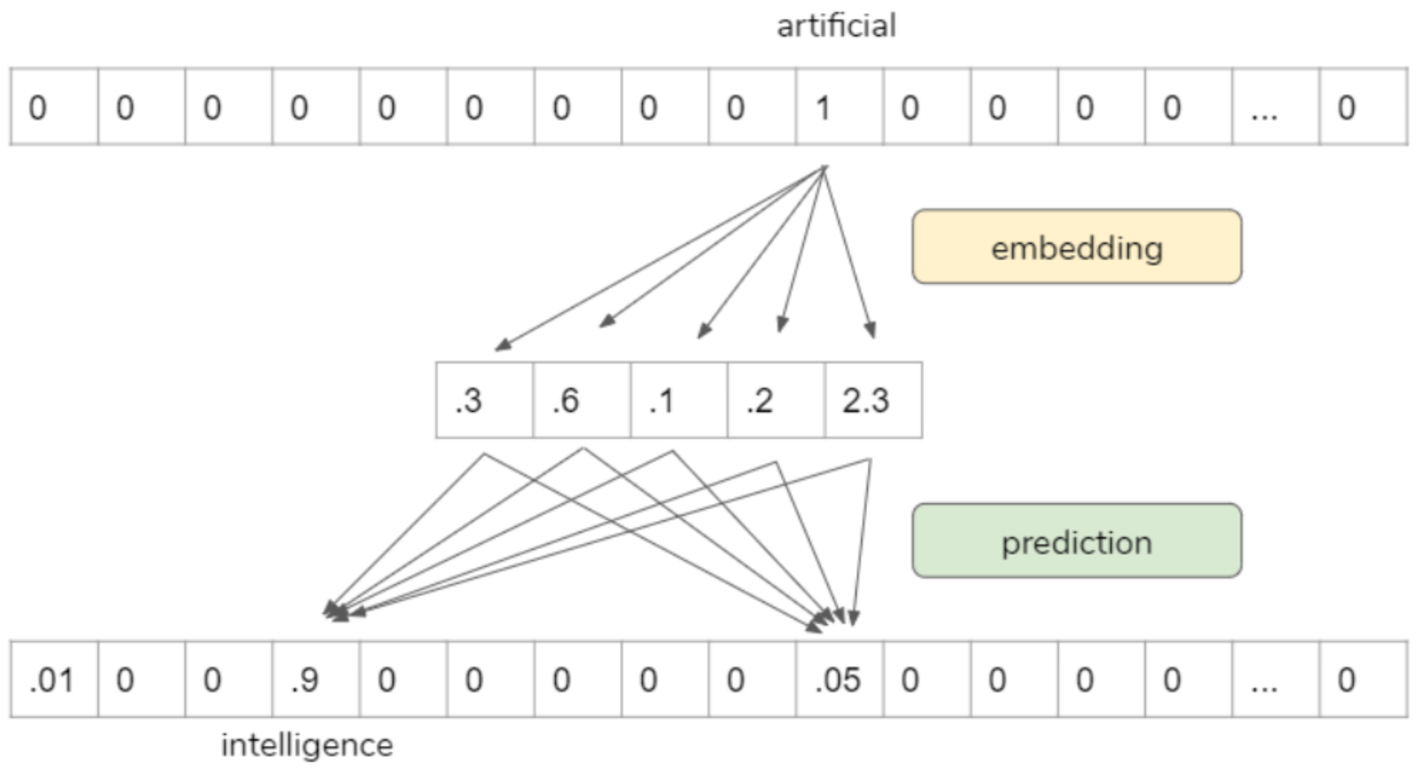
The basic idea of Word2vec is that instead of representing words as one-hot encoding (countvectorizer / tfidfvectorizer) in high dimensional space, we represent words in dense low dimensional space in a way that similar words get similar word vectors, so they are mapped to nearby points.

Word2vec is not deep neural network, it turns text into a numeric form that deep neural network can process as input.

How the word2vec model is trained

- Move through the training corpus with a sliding window: Each word is a prediction problem.
- The objective is to predict the current word using the neighboring words (or vice versa).
- The outcome of the prediction determines whether we adjust the current word vector. Gradually, vectors converge to (hopefully) optimal values.

For example, we can use "artificial" to predict "intelligence".

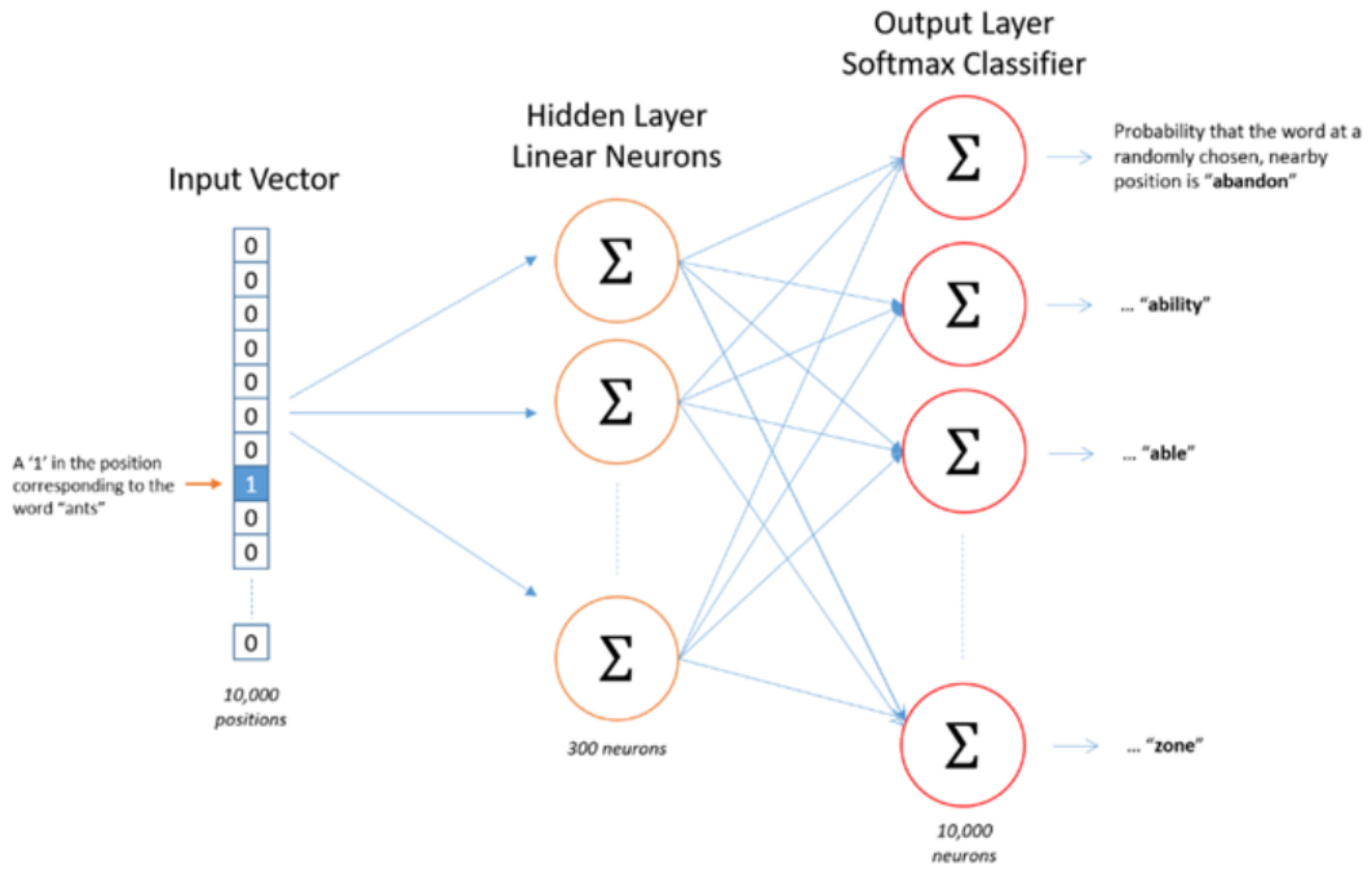


Source: https://www.infoq.com/presentations/nlp-practitioners/?itm_source=presentations_about_Natural-Language-Processing&itm_medium=link&itm_campaign=Natural-Language-Processing

However, the prediction itself is not our goal. It is a proxy to learn vector representations so that we can use it for other tasks.

Word2vec Skip-gram Network Architecture

This is one of word2vec models architectures. It is just a simple one hidden layer and one output layer.



Source: http://mccormickml.com/2016/04/19/word2vec-tutorial-the-skip-gram-model/

The Math

The following is the math behind word2vec embedding. The input layer is the one-hot encoded vectors, so it gets “1” in that word index, “0” everywhere else. When we multiply this input vector by weight matrix, we are actually pulling out one row that is corresponding to that word index. The objective here is to pull out the important row(s), then, we toss the rest.

$$\begin{bmatrix} 0 & 0 & 0 & 1 & 0 \end{bmatrix} \times \begin{bmatrix} 17 & 24 & 1 \\ 23 & 5 & 7 \\ 4 & 6 & 13 \\ 10 & 12 & 19 \\ 11 & 18 & 25 \end{bmatrix} = \begin{bmatrix} 10 & 12 & 19 \end{bmatrix}$$

Source: http://mccormickml.com/2016/04/19/word2vec-tutorial-the-skip-gram-model/

This is the main mechanics on how word2vec works.

When we use Tensorflow / Keras or Pytorch to do this, they have a special layer for this process called “Embedding layer”. So, we are not going to do math by ourselves, we only need to pass one-hot encoded vectors, the “Embedding layer” does all the dirty works.

Now we’re going to implement word2vec embedding for an already-prepared data set.

- We use python as our implementation language.
- Python provides us with a useful library for Natural Language Processing (NLP) called GenSim.
- Word2vec model is trained using text8 dataset (corpus). Text8 is preprocessed English corpus, which contains lots of words separated by space.

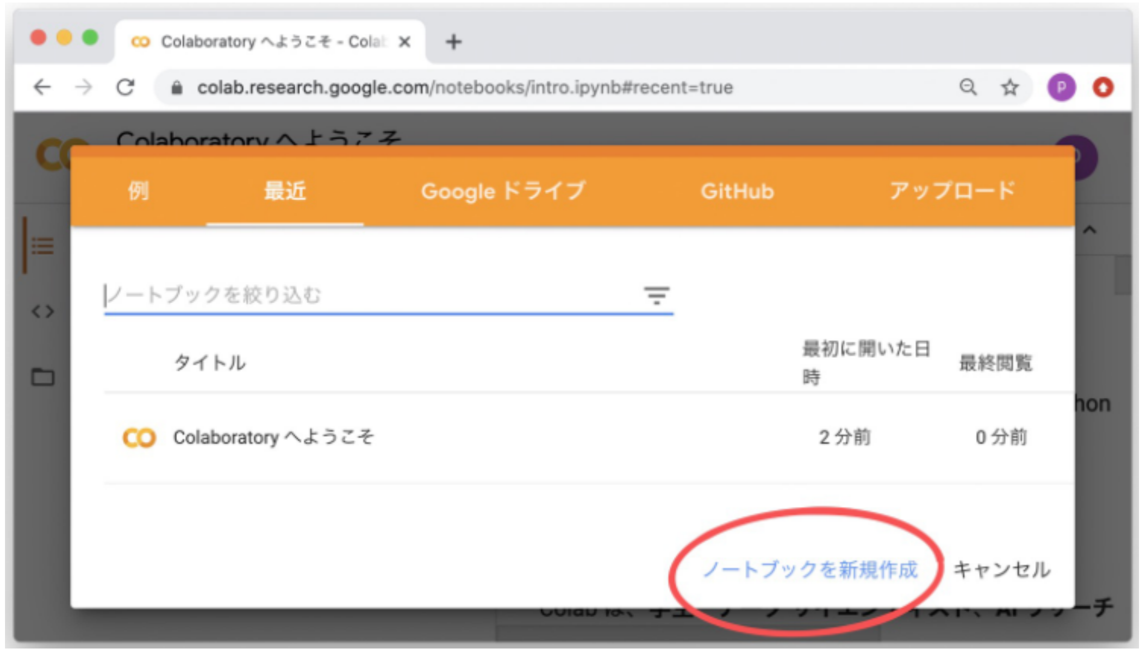
How to write and run a program in Python?

- You can download the latest version of python installer. After the installation of python, the following command allows you to use GenSim library:

```
> pip install gensim (hit Enter)
```

- But you may meet some difficulties when building the development environment. Considering this, one of the easiest way is to make and run a python program under cloud environment.

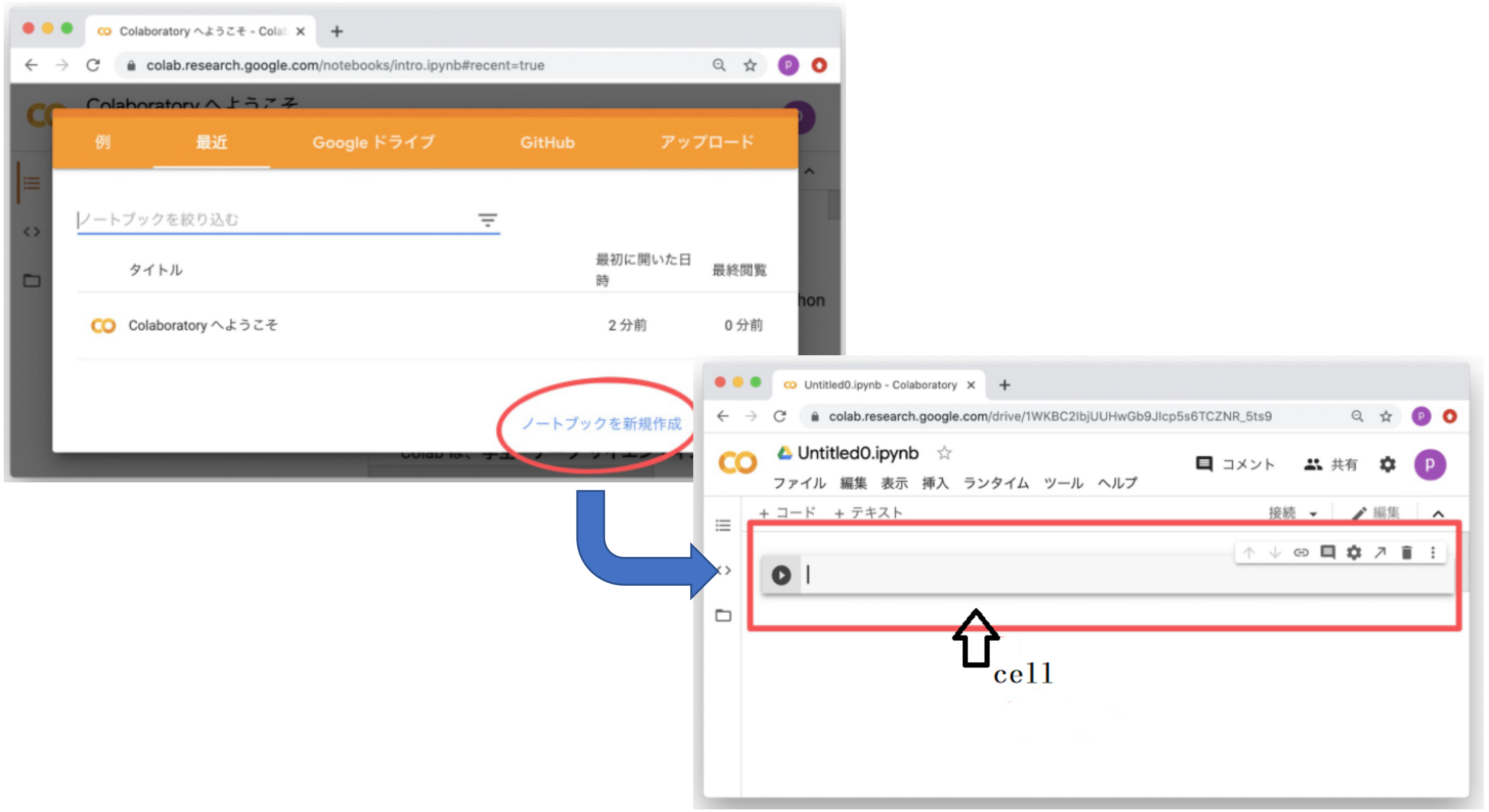
- Please check if you can access to Google Colaboratory (via VPN, if necessary).



https://colab.research.google.com/notebooks/intro.ipynb#recent=true

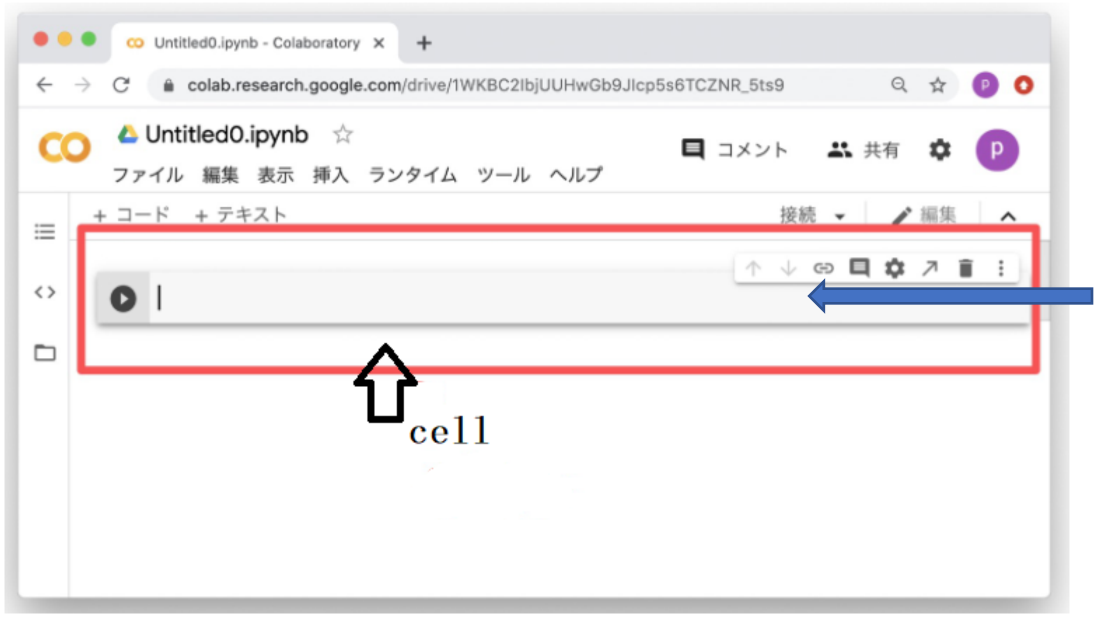
How to write and run a program in Python?

- Click on "ノートブックを新規作成" (red circle in the following figure) and open a cell, where you can type codes.

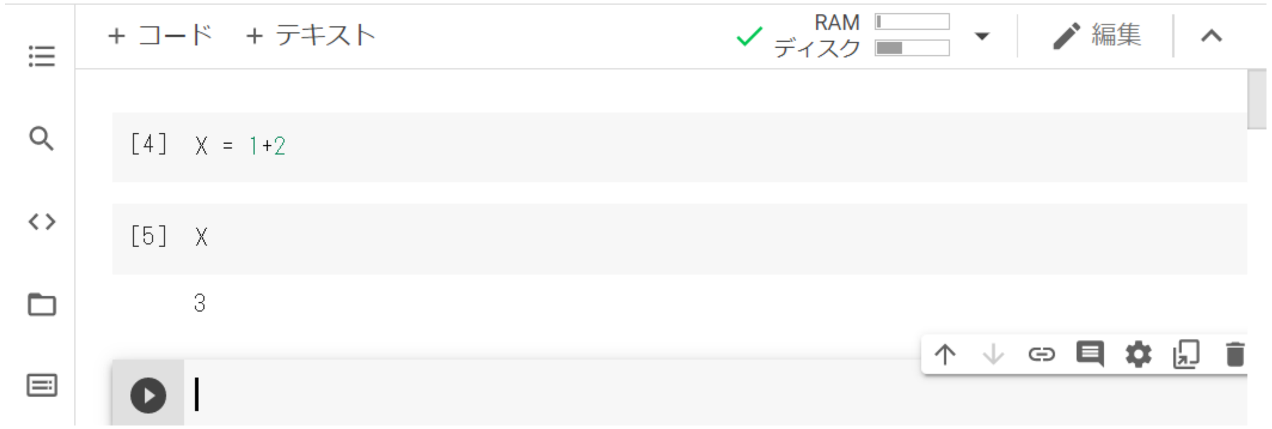


How to write and run a program in Python?

- Type the following statement and hit shift-enter.
  (first execution make take quite a long time)


Untitled0.ipynb - Colaboratory
colab.research.google.com/drive/1WKBC2IbjUUHwGb9Jlcp5s6TCZNR_5ts9
Untitled0.ipynb
ファイル 編集 表示 挿入 ランタイム ツール ヘルプ
コメント
共有
+ コード + テキスト
接続
編集
cell

```
X = 1+2
```

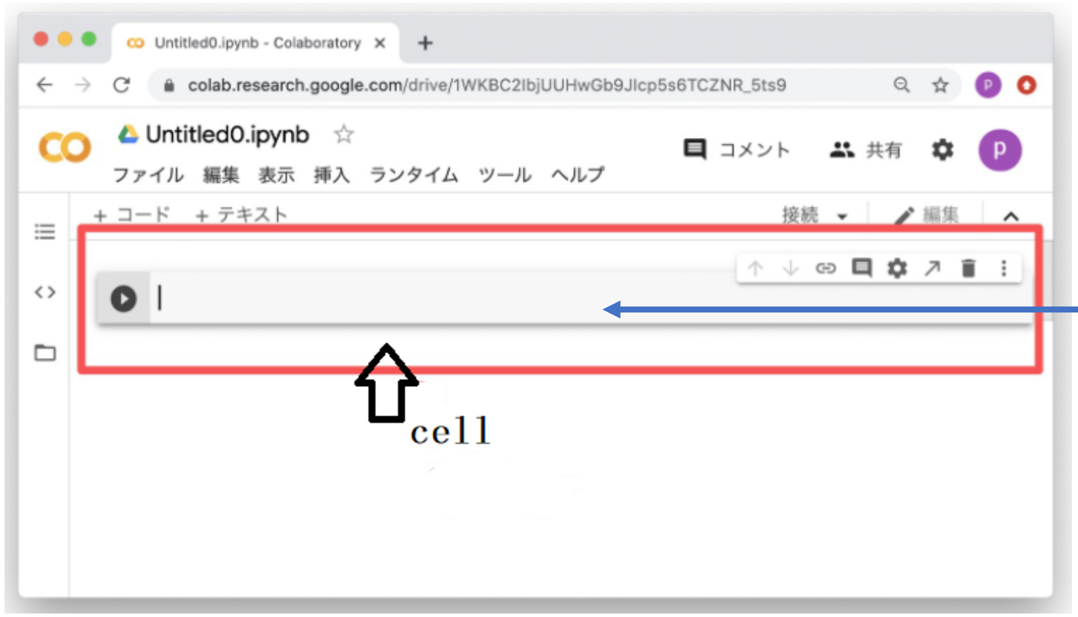


```
[4]  X = 1+2

[5]  X

     3
```

Training of word2vec model using GenSim library


Untitled0.ipynb - Colaboratory
colab.research.google.com/drive/1WKBC2IbjUUHwGb9Jlcp5s6TCZNR_5ts9
Untitled0.ipynb
ファイル 編集 表示 挿入 ランタイム ツール ヘルプ
コメント
共有
+ コード + テキスト
接続
編集
cell

STEP 1 – loading data set

```
import gensim.downloader as gendl
corpus = gendl.load("text8")
```

“test8” is the name of corpus (training data set).
It is loaded and stored in “corpus”.

STEP 2 - training

```
import gensim #ライブラリgensimを導入する
model = gensim.models.Word2Vec(corpus, size=5, window=2, iter=1, min_count=1)
```

length of context

iteration

NOTE
In some version of gensim, `size` is already replaced with `vector_size` and `iter` already replaced with `epoch`.

Similarity of words

STEP 3 – larger scale training

```
model = gensim.models.Word2Vec(corpus, size=200, window=5, iter=10, min_count=1)
```

NOTE
In some version of gensim, `size` is already replaced with `vector_size` and `iter` already replaced with `epoch`.

STEP 4 – similarity between words

```
model.wv.most_similar("coffee")
```

result

```
[('wool', 0.6894910335540771),
('cocoa', 0.6891734600067139),
('seafood', 0.6766855716705322),
('meat', 0.6657349467277527),
('cigarettes', 0.6616493463516235),
('wheat', 0.6599244475364685),
('syrup', 0.6596278548240662),
('cotton', 0.6591023206710815),
('cheese', 0.658471941947937),
('liquor', 0.6500911116600037)]
```

Interesting calculation of words

```python
model.wv.most_similar(positive=['queen', 'man'], negative=['woman'])
```

```
[('king', 0.4555075764656067),
('duke', 0.45153024792671204),
('lord', 0.44045498967170715),
('regent', 0.4297080338001251),
('crown', 0.42514702677726746),
('wales', 0.4120979309082031),
('victoria', 0.4099408984184265),
('portugal', 0.40795454382896423),
('aragon', 0.40479281544685364),
('prince', 0.4001534581184387)]
```

which means
queen – woman + man = king ?

Try other calculation and enjoy!