



Title	人文学研究者必見！テキストデータとTEIで描く新たな研究ビジョン
Author(s)	吉賀, 夏子; 田畑, 智司; 甲斐, 尚人 他
Citation	
Version Type	VoR
URL	https://doi.org/10.18910/101978
rights	This content is licensed under a Creative Commons Attribution-ShareAlike 4.0 International License.
Note	

The University of Osaka Institutional Knowledge Archive : OUKA

<https://ir.library.osaka-u.ac.jp/>

The University of Osaka

TEIハンズオン

応用編 「走れメロス」でテキスト分析

このセクションについて

TEIハンズオン基礎編までの知識を前提に、TEI形式に変換された研究対象のテキストを使った分析例をいくつか紹介します。

準備するもの

- demo(TEI-tools) > **TEI_hands-on.ipynb**
- Google Chromeブラウザ(推奨)



demo/TEI-tools(Github)は本動画の案内から入手できます。

Google Colab(Colaboratory, グーグルコラブ)
<https://colab.research.google.com/>

Google Colab (Colaboratory)の外観

<https://colab.research.google.com/?hl=ja>

The screenshot shows the Google Colab (Colaboratory) homepage in Japanese. The browser address bar displays `colab.research.google.com`. The page header includes the Colab logo, the text "Colaboratory へようこそ", and navigation links: "ファイル", "編集", "表示", "挿入", "ランタイム", "ツール", and "ヘルプ". A "ログイン" button is visible on the right. The left sidebar contains a "目次" (Table of Contents) with links to "はじめに", "データサイエンス", "機械学習", and "その他のリソース", along with a "使用例" (Usage Examples) section and a "+ セクション" button. The main content area features a "Colab へようこそ" (Welcome to Colab) heading, followed by "(新規) Gemini API をお試しください" (Try the new Gemini API). Below this is a list of links: "Generate a Gemini API key", "Talk to Gemini with the Speech-to-Text API", "Gemini API: Quickstart with Python", "Gemini API code sample", "Compare Gemini with ChatGPT", and "More notebooks". At the bottom, there is a paragraph in Japanese: "すでに Colab をよくご存じの場合は、この動画でインタラクティブなテーブルコードの履歴表示、コマンドパレットについてご覧ください。" (If you already know Colab well, please watch this video about interactive table code history display and command palette). Below the text is a video player thumbnail titled "3 Cool Google Colab Features" showing a man's face.

← → ↻ colab.research.google.com ☆ 📄 🗨️ 🗑️ 📄 | 🧑 一時停止中 ⋮

Colaboratory へようこそ ⚙️ 🔗 共有 ログイン

ファイル 編集 表示 挿入 ランタイム ツール ヘルプ

☰ 目次 🗑️ × + コード + テキスト ドライブにコピー 接続 ▼ ^

🔍 はじめに
{x} データサイエンス
🔑 機械学習
📁 その他のリソース
使用例
+ セクション

Colab へようこそ

(新規) Gemini API をお試しください

- [Generate a Gemini API key](#)
- [Talk to Gemini with the Speech-to-Text API](#)
- [Gemini API: Quickstart with Python](#)
- [Gemini API code sample](#)
- [Compare Gemini with ChatGPT](#)
- [More notebooks](#)

すでに Colab をよくご存じの場合は、この動画でインタラクティブなテーブルコードの履歴表示、コマンドパレットについてご覧ください。

3 Cool Google Colab Features

Google Colab (Colaboratory)?

ウェブブラウザ上で使えるお手軽プログラミングツール

- パソコンのOS(オペレーティングシステム、MacOSやWindows等)の違いや性能を気にする必要がなくなる
- 説明文をコードの前後に入力したり、トピックのアウトラインを作ることができる
- プログラミングの作成と実行の仕方がわかりやすい

<https://colab.research.google.com/?hl=ja>

- インターネットに接続し
Googleアカウントを作る必要がある
- プログラムを実行するには、Googleのコンピュータに接続するが長時間接続することができない
- 接続が切断されると一時的にインストールしたライブラリやデータは消える

Googleアカウントを作る

- Google Colabのサービスを使うにはGoogleの共通アカウント(ユーザ名とパスワードのセット)を作る必要があります。
- アカウントを作ってColabのプログラムを動かすだけであれば無料で使えます。
- アカウントがない場合は、次の動画に進む前に以下のページでご自分のアカウントを作ってください。
- <https://www.google.com/intl/ja/account/about/> (または「Googleアカウント作成」といったキーワードでインターネットを検索してください。)

Google Colabにログインしよう

The screenshot shows the Google Colab homepage. A red dashed border highlights the main content area. A red circle highlights the 'ログイン' (Login) button in the top right corner. A red arrow points to the 'ログイン' button with the text 'ここからログイン' (Login from here). The URL 'https://colab.research.google.com' is displayed in the browser's address bar. The page title is 'Colaboratory へようこそ' (Welcome to Colaboratory). The navigation bar includes links for 'ファイル' (Files), '編集' (Edit), '表示' (View), '挿入' (Insert), 'ランタイム' (Runtime), 'ツール' (Tools), and 'ヘルプ' (Help). The left sidebar shows a '目次' (Table of Contents) with links to 'はじめに' (Getting started), 'データサイエンス' (Data science), '機械学習' (Machine learning), 'その他のリソース' (Other resources), and '使用例' (Use cases). The main content area displays the URL 'https://colab.research.google.com' and the text 'ノートブックの画面' (Notebook screen). Below this, there is a list of links: 'Generate a Gemini API key', 'Talk to Gemini with the Speech-to-Text API', 'Gemini API: Quickstart with Python', 'Gemini API code sample', 'Compare Gemini with ChatGPT', and 'More notebooks'.

← → ↻ colab.research.google.com ☆

ここからログイン

Colaboratory へようこそ

ファイル 編集 表示 挿入 ランタイム ツール ヘルプ

目次

はじめに

データサイエンス

機械学習

その他のリソース

使用例

+ セクション

https://colab.research.google.com

ノートブックの画面

- [Generate a Gemini API key](#)
- [Talk to Gemini with the Speech-to-Text API](#)
- [Gemini API: Quickstart with Python](#)
- [Gemini API code sample](#)
- [Compare Gemini with ChatGPT](#)
- [More notebooks](#)

ノートブックを開こう

ファイル > ノートブックをアップロード > アップロード
を選んでダウンロードした”**TEI_hands-on.ipynb**”を
選択してください。このファイルはCLEから他のサンプル
ファイルとともにダウンロードできます。

ノートブックを開く

例 >

最近 >

Google ドライ
ブ >

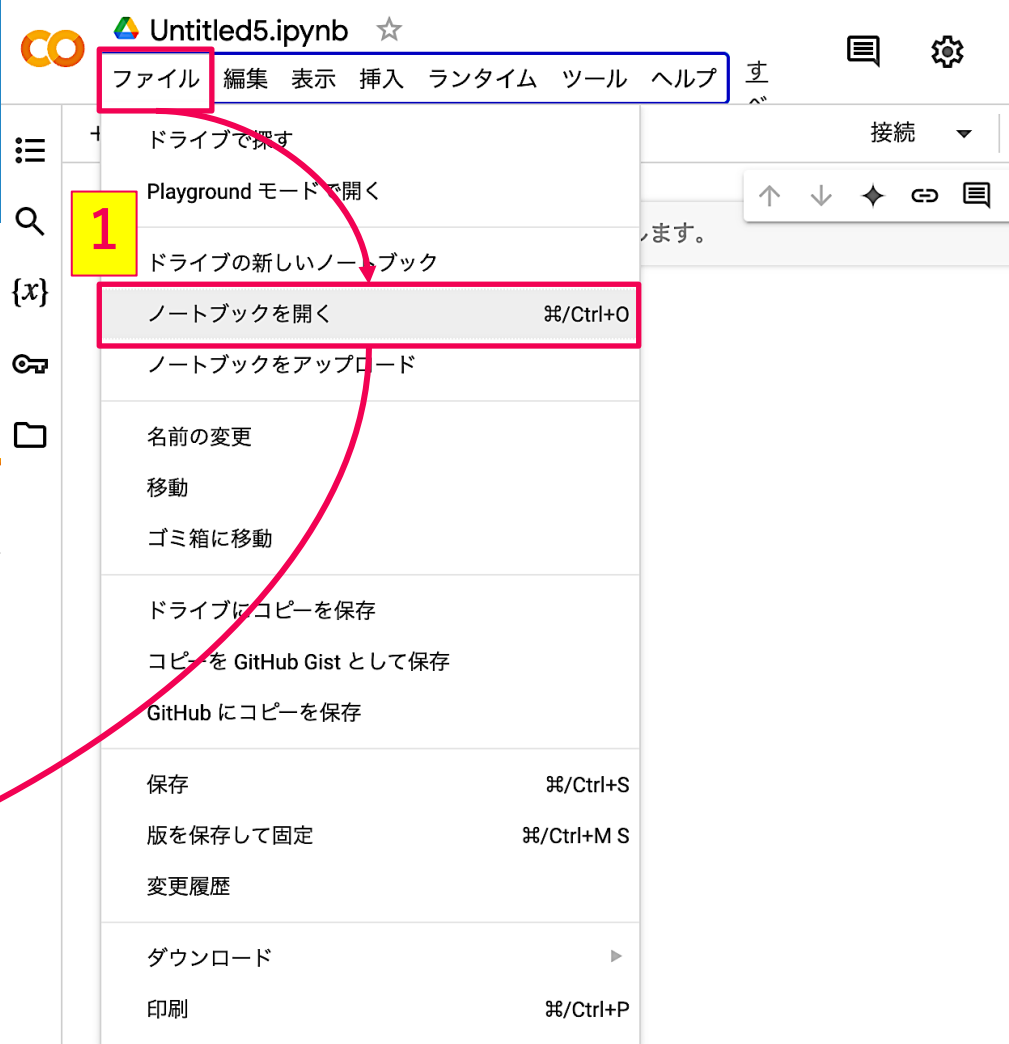
GitHub >

アップロード >

2

参照

または、ここにファイルをドラッグしてください



TEI_hands-on.ipynbについて

Google Colabで実際に動かせる、解説付きサンプルコード

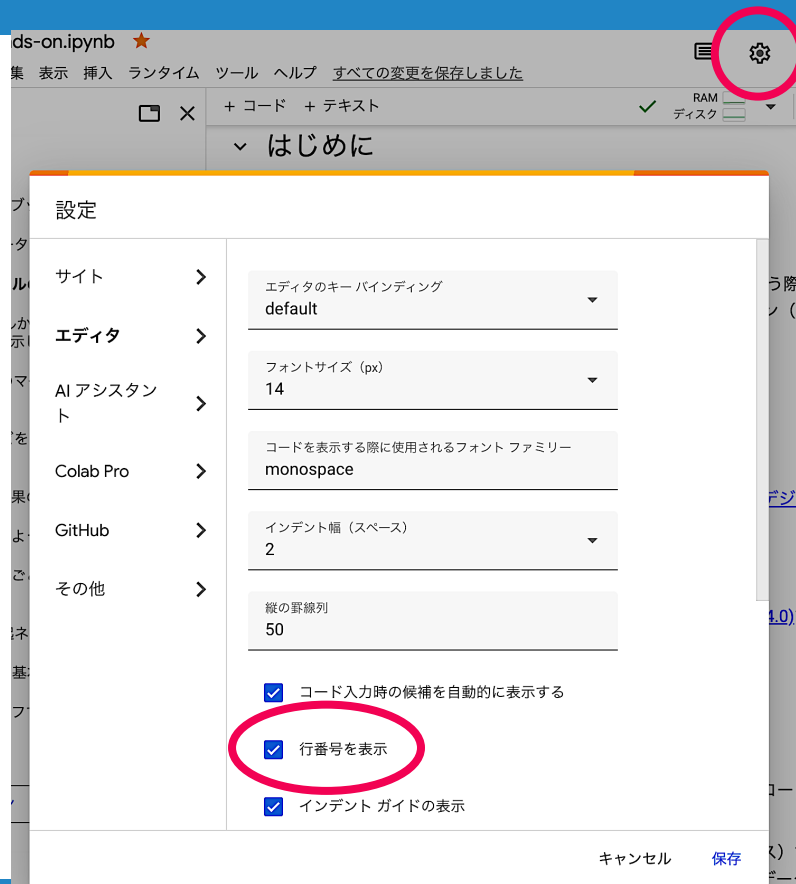
<https://colab.research.google.com/drive/1t0-N22TQzqjZ-cAVtFNKbb6hcZdQ7gbX?usp=sharing>

自分でコードを書き換える場合は、一旦コピーを保存してください。

サンプルコードはたくさんの事例があるPython言語で書かれています

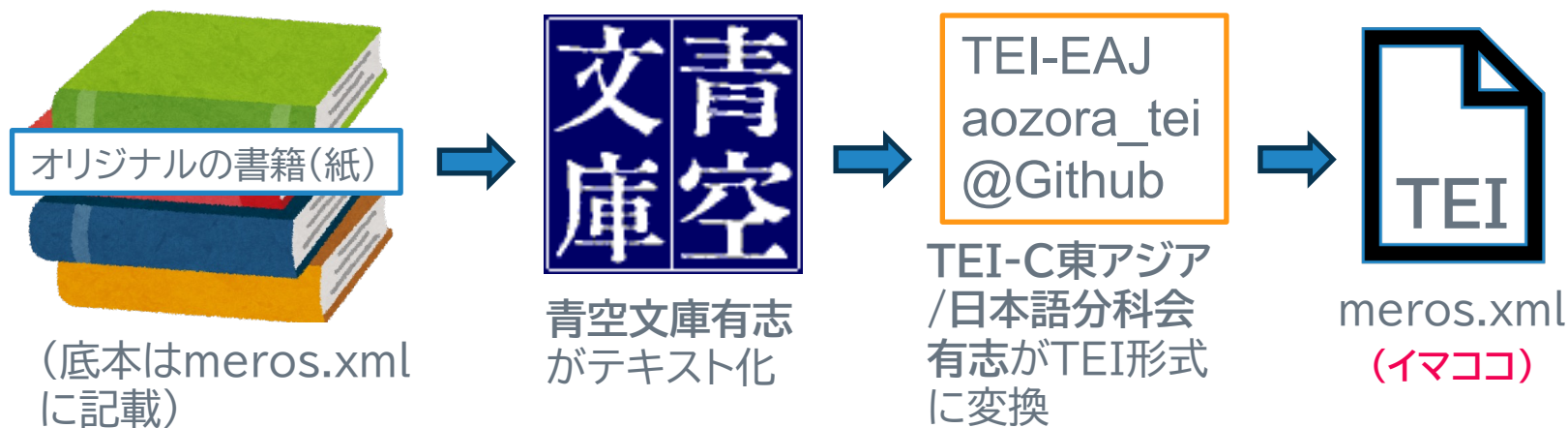
Google Colabの設定

行番号で説明できるように
歯車マークの設定
から、「エディタ」をクリ
ックして、「行番号を表
示」にチェックを入れて
ください。



太宰治「走れメロス」のmeros.xml

青空文庫のテキストを基にTEI化したサンプルに軽微な修正を行ったもの



環境設定とデータの読み込み

BeautifulSoup: PythonでXMLファイルを簡単に解析するためのライブラリ(まとまったプログラムのセット)のひとつ。任意のタグがついたデータを指定して抽出できる。



```
!pip install beautifulsoup4
```

実行ボタンを押すとこのコマンドを実行します



```
Requirement already satisfied: beautifulsoup4 in /usr/local/lib/python3.10/dist-packages  
Requirement already satisfied: soupsieve>1.2 in /usr/local/lib/python3.10/dist-packages (
```

meros.xmlをダウンロード

TEIデータをGithubというリポジトリ(基本無料のデータ格納庫)からダウンロードします。

押す→

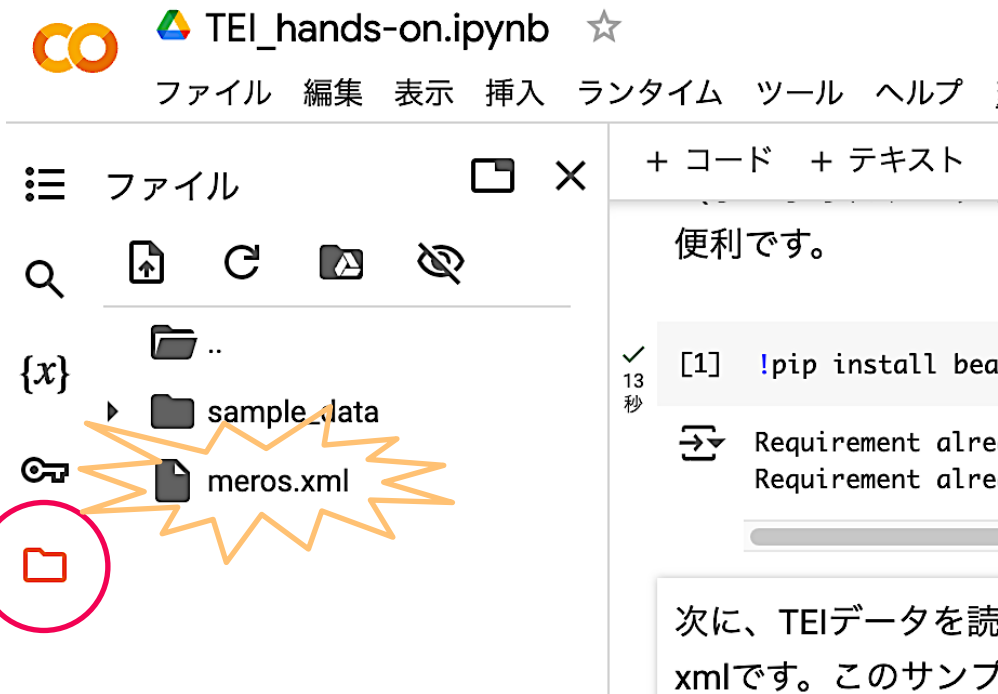


```
!curl -O 'https://raw.githubusercontent.com/nikolito/TEI-tools/main/meros.xml'
```



% Total	% Received	% Xferd	Average Speed	Time	Time	Time	Current
			Dload	Upload	Total	Spent	Speed
100 87937	100 87937	0 0	277k	0	--:--:--	--:--:--	277k

ダウンロードしたファイルはどこへ？



The screenshot shows the Google Colab interface for a notebook titled "TEI_hands-on.ipynb". The top menu bar includes "ファイル" (File), "編集" (Edit), "表示" (View), "挿入" (Insert), "ランタイム" (Runtime), "ツール" (Tools), and "ヘルプ" (Help). The left sidebar shows the "ファイル" (File) view with a search icon and a folder icon. The file explorer displays a directory structure with a folder named "sample_data" and a file named "meros.xml". A red circle highlights the folder icon in the sidebar, and a blue speech bubble points to it with the text "このフォルダアイコンをクリック" (Click this folder icon). The main area shows the code editor with a single cell containing the command `!pip install bea`. The output shows a success message: "Requirement already satisfied: bea [13 秒]". Below the output, a message says "次に、TEIデータを読むxmlです。このサンプル" (Next, read the TEI data in xml format. This sample).

TEI_hands-on.ipynb ☆

ファイル 編集 表示 挿入 ランタイム ツール ヘルプ

ファイル

+

コード

テキスト

便利です。

✓ [1] !pip install bea
13 秒

Requirement already satisfied: bea [13 秒]

次に、TEIデータを読むxmlです。このサンプル

このフォルダ
アイコンを
クリック

meros.xmlの中身

Pythonでよく使われるプログラムパターン:
ファイルの入出力

```
with open("meros.xml",  
encoding='utf8') as f:  
    print(f.read())
```

```
▶ with open("meros.xml", encoding='utf8') as f:  
    print(f.read())
```

```
⇒ <?xml version="1.0" encoding="UTF-8"?>  
  <TEI xmlns="http://www.tei-c.org/ns/1.0">  
    <teiHeader>  
      <fileDesc>  
        <titleStmt>  
          <title>走れメロス</title>  
          <author>太宰治</author>  
          <respStmt>  
            <resp>Transcription</resp>  
            <name>金川一之</name>  
          </respStmt>  
          <respStmt>  
            <resp>Proofreading</resp>  
            <name>高橋美奈子</name>  
          </respStmt>  
          <respStmt>  
            <resp>TEI encoding</resp>  
            <name>永崎研宣</name>  
          </respStmt>  
        </titleStmt>  
        <publicationStmt>  
          <distributor>青空文庫</distributor>  
          <authority>金川一之</authority>  
          <authority>高橋美奈子</authority>  
          <date when="2011-01-17"> 2011年1月17日</date>  
        </publicationStmt>  
        <sourceDesc>  
          <bibl>  
            <author>太宰治</author>  
            <title>走れメロス</title>
```

出力結果の表示/非表示の切り替え

このマークから
選択できます→



```
with open("meros.xml", encoding='utf8') as f:
    ... print(f.read())
```

出力を表示 / 非表示

選択した出力を消去

出力を全画面表示

メロス</title>
<author>太宰治</author>
<respStmt>
<resp>Transcription</i

TEIファイルからタグを除去したテキストを表示 1

- file_path、soupなどの文字列は**変数**
 - 変数はデータに文字列の名前がついた一時的な入れ物
- **=**の右側にある値をデータとして入れる
- **関数**は何らかの値をカッコ内に入れると、値に応じた結果を出力するもの

```
[ ] 1  #モジュールのインストール
2  from bs4 import BeautifulSoup
3
4  # TEIファイルを読み込んで、beautifulsoupがxml要素を取り出す
5  file_path = "meros.xml"
6  変数
7  with open(file_path, "r", encoding="utf-8") as file:
8  | | soup = BeautifulSoup(file, "xml") 関数
9
10 # <rt>タグを削除する
11 for ruby in soup.find_all("rt"):
12 | | ruby.decompose()
13
14 # <text>要素を取り出して、その中のタグを除去（タグなしの本文テキストにする）
15 text_content = soup.find("text").get_text(strip=True)
16
17 # 最初の1000文字をタグなしテキストから表示する
18 print(text_content[:1000] + "...")
```

TEIファイルからタグを除去したテキストを表示 2

- ルビを表示するタグのうち、rt要素を全て抽出する関数:
`find_all("rt")`

ルビの例)

```
<ruby>
  <rb>邪智暴虐</rb>
  <rt>じゃちぼうぎゃ</rt>
</ruby>
```

冒頭文「メロスは激怒した。」

0 1 2 3 4 5 6 7 8

変数の中に入っているテキストデータには0から番号が振られている

```
[ ] 1 #モジュールのインストール
    2 from bs4 import BeautifulSoup
    3
    4 # TEIファイルを読み込んで、beautifulsoupがxml要素を取り出す
    5 file_path = "meros.xml"
    6
    7 with open(file_path, "r", encoding="utf-8") as file:
    8     | soup = BeautifulSoup(file, "xml")
    9
   10 # <rt>タグを削除する
   11 for ruby in soup.find_all("rt"):
   12     | ruby.decompose()
   13
   14 # <text>要素を取り出して、その中のタグを除去 (タグなしの本文テキストにする)
   15 text_content = soup.find("text").get_text(strip=True)
   16
   17 # 最初の1000文字をタグなしテキストから表示する
   18 print(text_content[:1000] + "...")
```

textwrapライブラリを使ってもう少し読みやすく整形



```
1 # 20文字で折り返し、300字まで表示してみる
```

```
2 import textwrap
```

```
3
```

```
4 wrapped_text = textwrap.fill(text_content, width=20)
```

```
5 print(wrapped_text[:300] + "...")
```

```
6
```

スライス記法

0番目から299番目までの文字列(=300文字分)のwrapped_text

textwrapのライブラリに含まれる**fill関数**を使ってtext_contentを幅20文字かつ300文字まで折り返す



メロスは激怒した。必ず、かの邪智暴虐の王を除かなければならぬと決意した。メロスには政治がわからぬ。メロスは、村の牧人である。

笛を吹き、羊と遊んで暮して来た。けれども邪悪に対しては、人一倍に敏感であった。きょう未明メロスは村を出発し、野を越え山越え、十里はなれた此のシラクスの市にやって来た。メロスには父も、母も無い。女房も無い。十六の、内気な妹と二人暮しだ。この妹は、村の或る律気な一牧人を、近々、花婿として迎える事になっていた。結婚式も間近かなのである。メロスは、それゆえ、花嫁の衣裳やら祝宴の御馳走やらを買いに、はるばる市にやって来たのだ。 先ず、その品々を買い集...

このファイルのマークアップルールを確認するには

- どんなタグが使われているか？
- タグの階層構造はどうなっているのか？

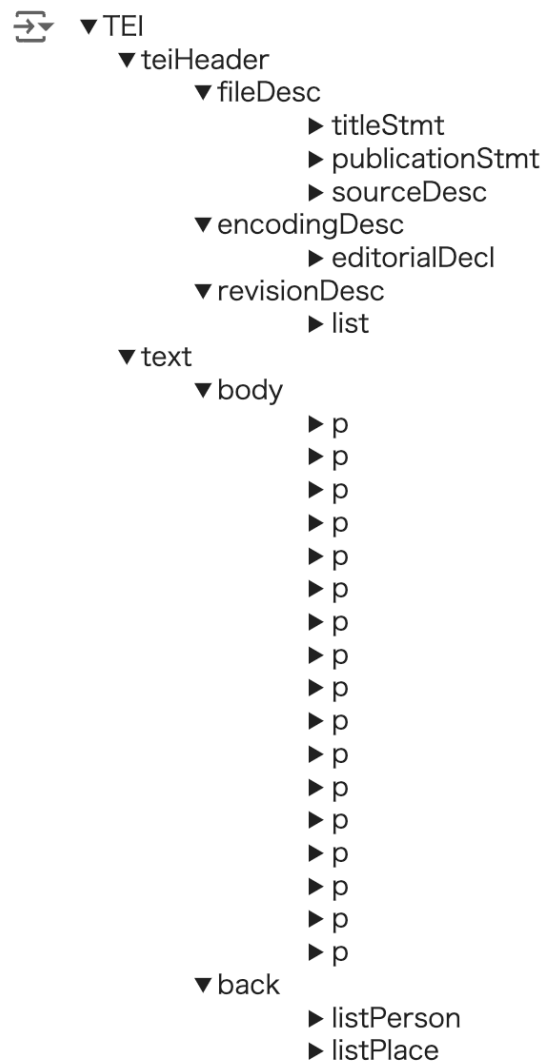


```
1 from bs4 import BeautifulSoup
2 from IPython.display import display, HTML
3
4 def generate_collapsible_html(tag, indent=0):
5     # 再的にxmlタグを階層表示します
6     children = tag.find_all(recursive=False)
7     if not children:
8         return f'<div style="margin-left: {indent}
9         px;">{tag.name}</div>'
10
11     inner_html = "".join
12     ([generate_collapsible_html(child, indent +
13     20) for child in children])
14     return f'<div style="margin-left: {indent}px;">
15     <summary>{tag.name}</summary>
16     {inner_html}
17     </div>
18     ""
19
20 # XML要素をsoupに入れます
21 file_path = "meros.xml"
22 with open(file_path, "r", encoding="utf-8") as
23     file:
24         soup = BeautifulSoup(file, "xml")
25
26 # HTML表示でタグの階層を開閉します
27 root = soup.find(True) # 最初のタグを取得
28 html_output = generate_collapsible_html(root)
29 display(HTML(html_output))
```

meros.xmlのタグ階層構造

XML形式のデータは必ず入れ子になっている

1. これまでの説明の通り、TEIタグの中にはteiHeader、textタグがある
2. teiHeaderにはfileDesc、encodingDesc、revisionDescタグがある
3. textタグにはbody(本文)およびback(後付けや補遺など本文以外の要素)がある
 - bodyにはpタグが17個ある
 - backにはlistPersonおよびlistPlaceタグがあり、それぞれ物語に登場する人物や地名の名称が入っている



editorialDeclタグの調査

editorialDeclとは:

マークアップルールを記すタグ

出力結果

1. **placeName**の@typeでは、実際にいた場所を”real”、そうでない場所を”unreal”として区別している
2. **persName**は、人称代名詞以外の人や人々を指す名詞に付与している
3. 人称代名詞には**rs**を付与している
4. (上記いずれのタグも)**@corresp**でID参照している



```
1  #モジュールのインストール
2  from bs4 import BeautifulSoup
3  import textwrap
4
5  # TEIファイルを読み込んで、beautifulsoupがxml要素
   # を取り出す
6  file_path = "meros.xml"
7  with open(file_path, "r", encoding="utf-8") as
   file:
8      | soup = BeautifulSoup(file, "xml")
9
10 # マークアップルールを記した<editorialDecl>を選択
    # して、変数editorialDeclに代入
11 editorialDecl = soup.select('editorialDecl')
12
13 # editorialDeclの中身を出力する
14 wrapped_text = textwrap.fill(editorialDecl[0].
   get_text(strip=True), width=50) # widthでお好みの
    幅に変えてください
15 print(wrapped_text.replace("  ", ""))
16
```

名寄せをしよう

```
1 from bs4 import BeautifulSoup
2 from collections import Counter
3
4 # TEIファイルを読み込んで、BeautifulSoupで解析
5 file_path = "meros.xml" # 実際のパスを指定
6 with open(file_path, "r", encoding="utf-8") as
  file:
7     soup = BeautifulSoup(file, "xml")
8
9 # <persName>タグのうちcorresp="#メロス"に該当する
  要素を抽出
10 melos_names = [pers.text for pers in soup.find_all(
  ("persName", {"corresp": "#メロス"})]]
11
12 # 呼び名とその出現回数をカウント
13 name_counts = Counter(melos_names)
14
15 # 結果を表示
16 for name, count in name_counts.items():
17     print(f"{name}: {count}回")
18
```

結果→



メロス: 68回

メロス: 6回

下賤
げせん
の者: 1回
嘘つき: 1回
兄: 4回
メロス ほどの男: 1回
おまえの兄: 1回
たぶん偉い男: 1回
真の勇者、メロス: 1回

おかしい箇所

自分: 1回

よくよく不幸な男: 1回
裏切者: 1回
地上で最も、不名誉の人種: 1回
悪徳者: 1回
醜い裏切り者: 1回
正義の士: 1回
正直な男: 2回
勇者: 1回

名寄せをしよう(改良版)

```
[ ] 1 from bs4 import BeautifulSoup
    2 from collections import Counter
    3
    4 # TEIファイルを読み込んで、BeautifulSoupで解析
    5 file_path = "meros.xml"
    6 with open(file_path, "r", encoding="utf-8") as file:
    7     soup = BeautifulSoup(file, "xml")
    8
    9 # <persName>タグのうちcorresp="#メロス"に該当する
   10 要素を抽出
   11
   12 melos_names = []
   13
   14 for pers in soup.find_all("persName", {"corresp":
   15     "#メロス"}):
   16     # <rb>タグがある場合はその内容を抽出、ない場合
   17     # は全体のテキストを使用
   18     if pers.find("rb"):
   19         melos_names.append(''.join(rb.text for rb
   20             in pers.find_all("rb")).strip())
   21     else:
   22         melos_names.append(pers.text.strip())
   23
   24 # 呼び名とその出現回数をカウント
   25 name_counts = Counter(melos_names)
   26
   27 # 結果を表示
   28 for name, count in name_counts.items():
   29     print(f"{name}: {count}回")
```

ルビの例)

```
<ruby>
  <rb>邪智暴虐</rb>
  <rt>じゃちぼうぎゃく</rt>
</ruby>
```



メロス: 74回
下賤: 1回
嘘つき: 1回
兄: 4回
メロス ほどの男: 1回
おまえの兄: 1回
たぶん偉い男: 1回
真の勇者、メロス: 1回
自分: 1回
よくよく不幸な男: 1回
裏切者: 1回
地上で最も、不名誉の人種: 1回
悪徳者: 1回
醜い裏切り者: 1回
正義の士: 1回
正直な男: 2回
勇者: 1回

パラグラフごとにメロスの呼称を観察しよう



```
1 from bs4 import BeautifulSoup
2 from collections import defaultdict
3 from IPython.display import display, Markdown
4
5 # TEIファイルを読み込んで、BeautifulSoupで解析
6 file_path = "meros.xml" # 実際のパスを指定
7 with open(file_path, "r", encoding="utf-8") as
  file:
8     | soup = BeautifulSoup(file, "xml")
9
10 # text > body > pタグについて、各<p>タグごとにメロスの呼び名を抽出する
11 paragraphs = soup.find("text").find("body").
  find_all("p")
12 melos_names_by_paragraph = defaultdict(list)
13
14 for i, p in enumerate(paragraphs, start=1):
15     | for pers in p.find_all("persName",
16     |   {"corresp": "#メロス"}):
17         | if pers.find("rb"):
18             | name = ''.join(rb.text for rb in pers.
19             |   find_all("rb")).strip()
20         | else:
21             | name = pers.text.strip()
22         | melos_names_by_paragraph[i].append(name)
23
24 # 結果を表示
25 for para_num, names in melos_names_by_paragraph.
  items():
26     | display(Markdown(f"### Paragraph {para_num}"))
27     | display(Markdown(", ".join(names) if names
28     |   else "--"))
```

Paragraph 1

メロス, メロス, メロス

Paragraph 2

メロス, メロス, メロス

Paragraph 3

メロス

Paragraph 4

メロス, メロス, メロス, メロス

Paragraph 5

メロス, メロス, メロス, メロス, メロス, メロス, 下賤,
メロス, メロス, 嘘つき, メロス

Paragraph 6

メロス, メロス, メロス

Paragraph 7

メロス, メロス, 兄, 兄, 兄, メロス, メロス, メロス, メ
ロス

Paragraph 8

メロス, メロス, メロス, メロス ほどの男, おまえの兄,
兄, たぶん偉い男, メロス, メロス, メロス

Paragraph 9

メロス, メロス

Paragraph 10

メロス, メロス, メロス

Paragraph 11

メロス, メロス, メロス, メロス, メロス, メロス

Paragraph 12

メロス

Paragraph 13

メロス, メロス, 真の勇者、メロス, 自分, よくよく不
幸な男, 裏切者, 地上で最も、不名誉の人種, 悪徳者,
醜い裏切り者

Paragraph 14

メロス, メロス, メロス, 正義の士, 正直な男, 正直な
男

Paragraph 15

メロス, メロス, メロス

Paragraph 16

メロス, メロス, メロス, メロス, メロス, メロス, メロス

Paragraph 17

メロス, メロス, メロス, メロス, メロス, メロス, メロス,
メロス, メロス, メロス, メロス, メロス, 勇者

メロス、ディオニス、セリヌンティウスの呼称を パラグラフごとに抽出

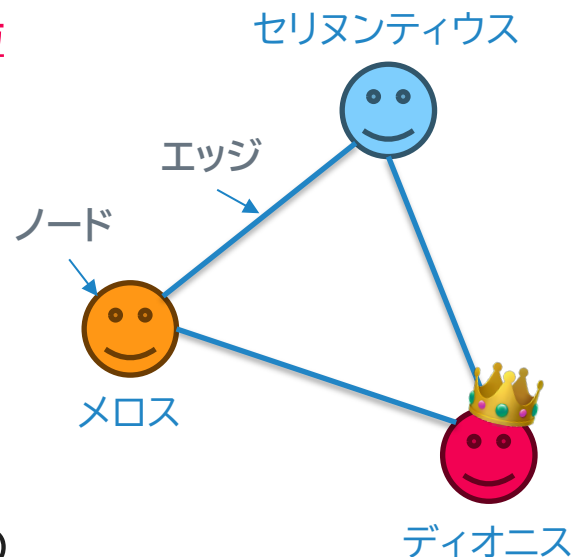
index	メロス	ディオニス	セリヌンティウス	Text (first 100 chars)
P1	メロス、メロス、 メロス	邪智暴虐		メロスは激怒した。必ず、かの邪智暴虐じやちぼうぎやくの王を除かなければならぬと決意した。メロスには政治がわからぬ。メロスは、村の牧人である。笛を吹き、
P2	メロス、メロス、 メロス			きょう未明メロスは村を出発し、野を越え山越え、十里はなれた此このシラクスの市にやって来た。メロスには父も、母も無い。女房も無い。十六の、内気な妹と二人暮しだ。この妹は、村の或る律気な一牧人を、近々、
P3	メロス		竹馬の友、セリヌンティウス	メロスには竹馬の友があった。セリヌンティウスである。今は此のシラクスの市で、石工をしている。その友を、これから訪ねてみるつもりなのだ。久しく逢わなかつ
P4	メロス、メロス、 メロス、メロス	王、王様、御自身、国王、王		歩いているうちにメロスは、まちの様子を怪しく思った。ひっそりしている。もう既に日も落ちて、まちの暗いのは当りまえだが、けれども、 なんだか、夜のせいばかりでは無く、市全体が、やけに寂しい。のんきなメ
P5	メロス、メロス、 メロス、メロス、 メロス、メロス、 下賤、メロス、メ ロス、嘘つき、メ ロス	王、ディオニス、 王、暴君、 王、王、暴君、 王、王、暴君、 わし	セリヌンティウス、無二の 友人、あの友人、身代りの 男、身代りの男、身代り、 その身代り	メロスは、単純な男であった。買い物、背負ったままで、 のそのそ王城にはいって行った。たちまち彼は、巡邏じゅんらの警吏に捕縛された。調べられて、メロスの懐中からは 短剣が出て来たので、

登場人物の共起ネットワーク分析

作品中で特定の人物がどの程度一緒に登場するか(共起)をパラグラフごとに分析する

手順:

- 1.各パラグラフで登場するキャラクター(<persName>)を抽出
- 2.パラグラフ単位で共に登場するキャラクターidのペアを集計
- 3.**ネットワークグラフ**を作成し、共起頻度をエッジの太さで表現



(ネットワーク)グラフ: 点(ノード)と線(エッジ)で何らかの関係を表現するデータ

登場人物の共起ネットワーク分析の準備

準備

必要なライブラリと日本語フォントを読み込む

主要ライブラリ

- **NetworkX**: Pythonでグラフ(ネットワーク)構造を扱うための強力なライブラリ。ノード(点)とエッジ(線)で構成されるネットワークを作成、操作、分析、そして可視化するために設計される。
- **matplotlib**: Pythonでデータの可視化を行うための代表的なライブラリ。特に、2Dグラフの作成に優れており、シンプルな線グラフから複雑なヒートマップや3Dプロットまで、多様なグラフ図を描画できる。

物語中の登場人物を抽出



```
1 # TEIファイルを読み込んでBeautifulSoupで解析
2 file_path = "meros.xml" # 実際のパスを指定
3 with open(file_path, "r", encoding="utf-8") as
  file:
4     soup = BeautifulSoup(file, "xml")
5
6 # listPersonから登場人物IDと名前を抽出
7 all_characters = {}
8 for person in soup.find_all("person"):
9     person_id = person.get("xml:id")
10    pers_name = person.find("persName")
11    if person_id and pers_name:
12        name = ''.join(rb.text for rb in
13            pers_name.find_all("rb")).strip() if
14            pers_name.find("rb") else pers_name.text.
15            strip()
16        all_characters[person_id] = name
17
18 for character in all_characters:
19     print(f"{character}: {all_characters[character]}")
20 "
```

(meros.xmlより)

```
<listPerson>
  <person xml:id="メロス">
    <persName>メロス</persName>
    <occupation>牧人</occupation>
  </person>
```

メロス: メロス
ディオニス: ディオニス
セリヌンティウス: セリヌンティウス
メロスの妹: メロスの妹
メロスの妹の婿: メロスの妹の婿
ディオニスの妹婿: 妹婿さま
ディオニスの妹: 妹さま
ディオニスの妹の御子: 御子さま
ディオニスの世嗣: お世嗣
アレキス: アレキス
老爺: 老爺
フィロストラトス: フィロストラトス

corresp属性にあるidで名寄せ

```
1 # パラグラフごとの共起データを収集
2 paragraphs = soup.find("text").find("body").
  find_all("p")
3 cooccurrence = defaultdict(list)
4
5 for para_num, p in enumerate(paragraphs, start=1):
6     present_characters = set()
7
8     for pers in p.find_all("persName"):
9         corresp = pers.get("corresp")
10        if corresp is not None:
11            corresp = corresp.strip("#")
12            if corresp in all_characters:
13                present_characters.add(corresp)
14
15 # 共起をパラグラフ別に記録
16 for char1 in present_characters:
17     for char2 in present_characters:
18         if char1 != char2:
19             cooccurrence[(char1, char2)].
20             append(para_num)
21
22 print(f"共起ペアの数: {len(cooccurrence)}")
23 print("共起ペア : 共起したパラグラフ番号")
24 for occurrence in cooccurrence:
25     print(f"{occurrence}: {cooccurrence[occurrence]}")
26     "
```

共起ペアの数: 78

共起ペア : 共起したパラグラフ番号

('メロス', 'ディオニス'): [1, 4, 5, 6, 8, 9, 10, 12, 13, 16, 17]

('ディオニス', 'メロス'): [1, 4, 5, 6, 8, 9, 10, 12, 13, 16, 17]

('メロス', 'メロスの妹の婿'): [2, 7, 8]

('メロス', 'メロスの妹'): [2, 5, 7, 8]

('メロスの妹の婿', 'メロス'): [2, 7, 8]

('メロスの妹の婿', 'メロスの妹'): [2, 7, 8]

('メロスの妹', 'メロス'): [2, 5, 7, 8]

('メロスの妹', 'メロスの妹の婿'): [2, 7, 8]

('メロス', 'セリヌンティウス'): [3, 5, 6, 10, 11, 13, 15, 16, 17]

('セリヌンティウス', 'メロス'): [3, 5, 6, 10, 11, 13, 15, 16, 17]

('メロス', 'ディオニスの世嗣'): [4]

('メロス', 'ディオニスの妹'): [4]

('メロス', 'ディオニスの妹の御子'): [4]

('メロス', 'アレキス'): [4]

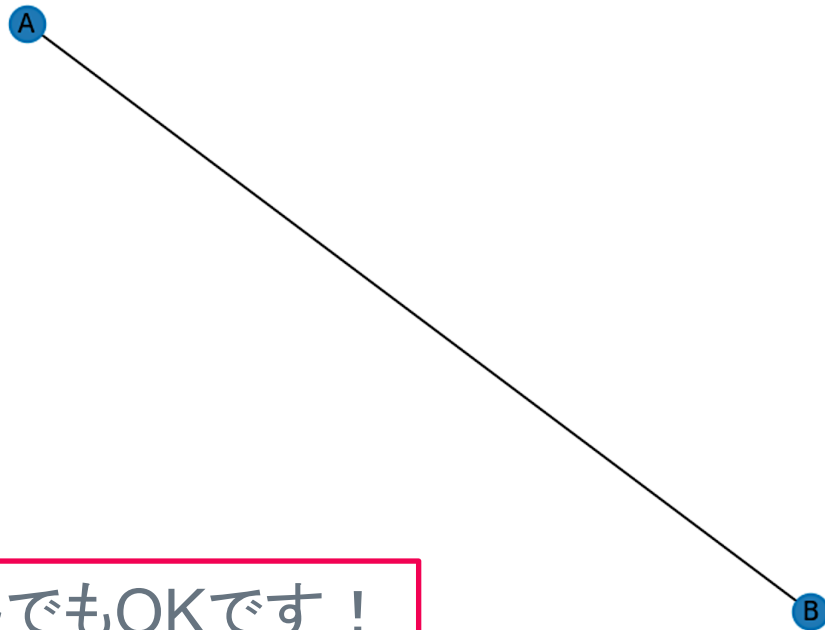
('メロス', 'ディオニスの妹の婿'): [4]

corresp属性を確かめれば、誰の名前なのかidでわかる

NetworkXの基本

Google Colab
「TEI_hands-on.ipynb」
の「NetworkXとは？」か
ら「コードの説明」までの項
目を直接参照してください。
ノード間にエッジを描く基本
を説明しています。

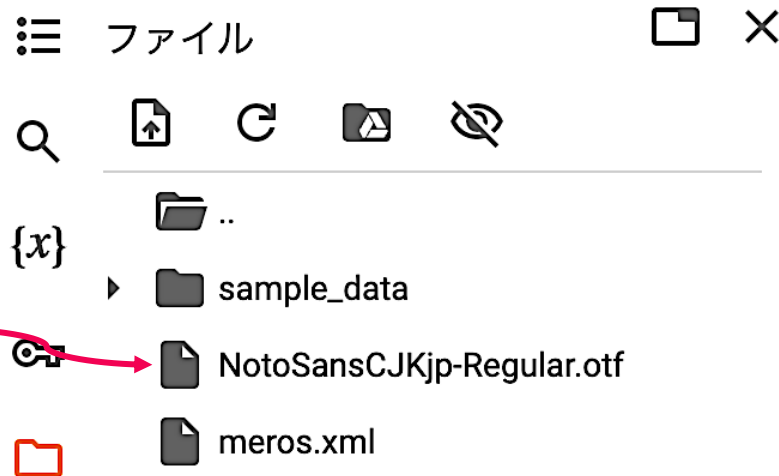
```
Nodes: ['A', 'B']  
Edges: [('A', 'B', {'weight': 3})]
```



一旦次の動画に進んでもOKです！

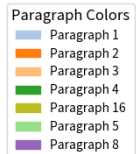
NetworkXを使ったパラグラフ別登場人物の共起の可視化 1

```
1 import networkx as nx
2 import matplotlib.pyplot as plt
3 from bs4 import BeautifulSoup
4 from collections import defaultdict
5 import matplotlib.font_manager as fm
6 import matplotlib.patches as mpatches
7
8 # 日本語フォントの設定 (Google Colabの場合には適切な
  # フォントを利用)
9 # ColabではIPAexGothicなどが利用可能
10 ! curl -O "https://raw.githubusercontent.com/
  notofonts/noto-cjk/main/Sans/OTF/Japanese/
  NotoSansCJKjp-Regular.otf"
11
12 font_path = 'NotoSansCJKjp-Regular.otf' # Update
  with your font file path
13 font_prop = fm.FontProperties(fname=font_path)
14 fm.fontManager.addfont(font_path)
15 plt.rcParams['font.family'] = font_prop.get_name()
```

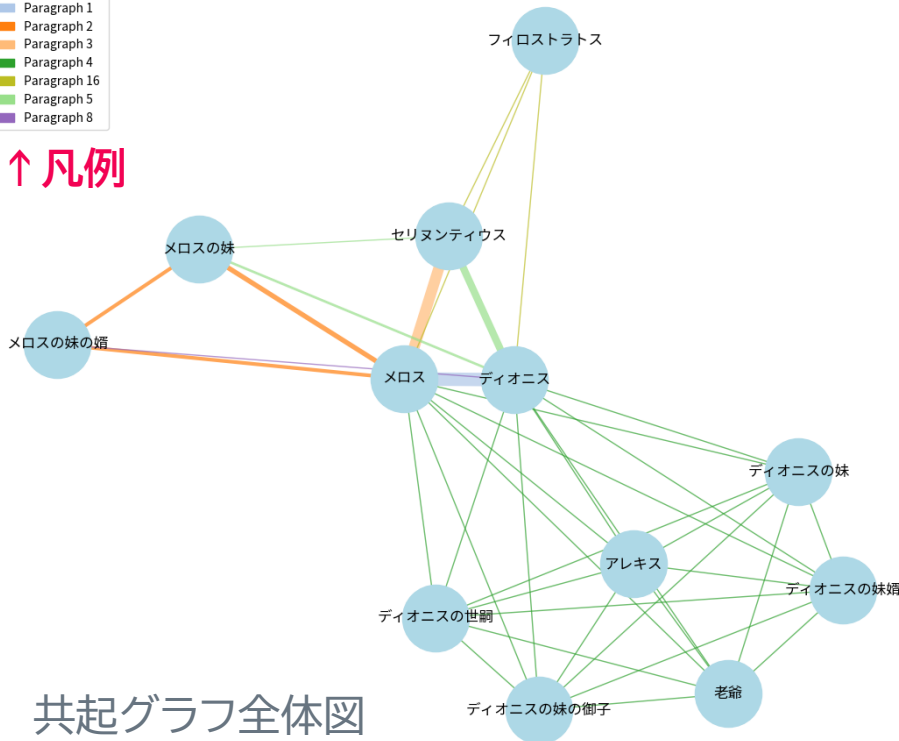


NetworkXを使ったパラグラフ別登場人物の共起の可視化 2

登場人物の共起ネットワーク (パラグラフ別色分け + エッジ太さは共起数)



↑ 凡例



共起グラフ全体図

【この図の読み方】

登場人物間を繋ぐエッジ

1. 太さ
2. 本数
3. 色の多さ

1-3が明確になることで、特定の登場人物同士の関係性の重要度、物語上での人物の重要度、物語の展開に伴う人間関係の変化がわかります。

より詳細な分析が必要な場合、各パラグラフの具体的な内容と照らし合わせて関係性を深掘りすることができます。

まとめ

- 「走れメロス」を使ったテキスト分析を紹介
- ノートブックはコピーして別の作品のTEIファイルに応用できる