TEI 2024

From Catullus to Wikidata: Language Models, Metadata Schemas, and Ontologies in a Digital Edition in TEI

Carlos Javier Nusch [1], Gabriel Alejandro Calarco [2], Gimena del Rio Riande [2], Leticia Cecilia Cagnina [2, 3], Marcelo Luis Errecalde [2, 3], Leandro Antonelli [4, 5]

[1] PREBI-SEDICI, UNLP, [2] CONICET, [3] LIDIC, UNSL,
[4] LIFIA, UNLP, [5] CAETI, UAI, Argentina.

carlosnusch@prebi.unlp.edu.ar

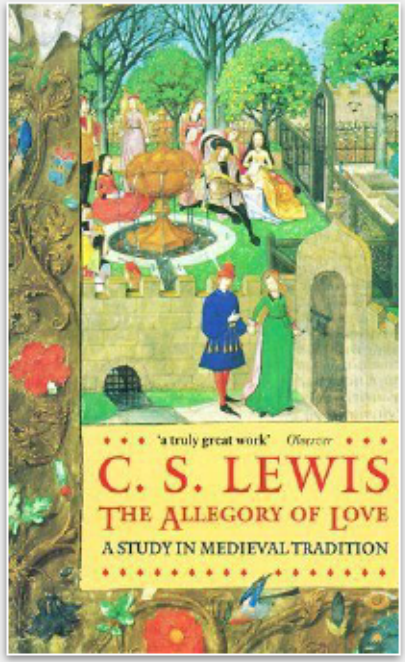
TEI 2024

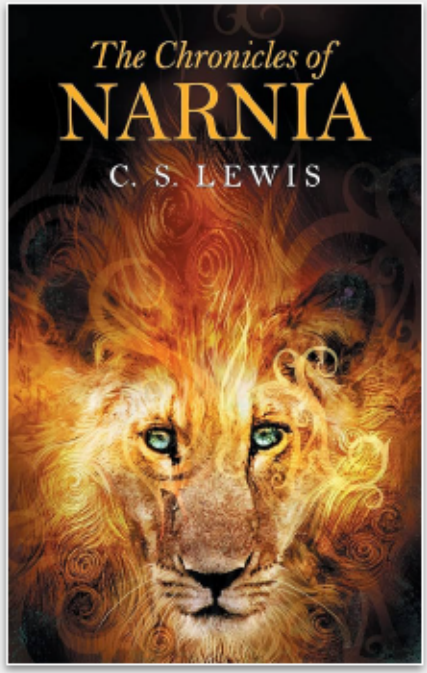
Aetatis Amoris Project: Markup and NLP for Latin Love Poetry

- This presentation describes the markup and **natural language processing** tasks undertaken in the *Aetatis Amoris* project, which explores love poetry and its textual patterns throughout literary history.
- In this early stage, the project focuses on the works of classical Latin poets such as *Catullus*, *Tibullus*, and *Propertius*.
- One approach involves using the **TEI-XML** standard to encode these texts.

Motivation: A Reading of C. S. Lewis

- **"French poets, in the eleventh century, discovered or invented, or were the first to express, that romantic species of passion which English poets were still writing about in the nineteenth."** (Lewis, 1936)
- Lewis suggested that the shape of the love motifs and imaginary we had, at least at the beginnings of the XX Century, belong to the occitan poets in the XI century.





A few minutes editing tasks

The first tasks involved automatically generating **key document elements** like the **header** and **body**, along with counting and **automatically tagging verses** and **stanzas**.

```python
from xml.etree.ElementTree import Element, SubElement, ElementTree
from pathlib import Path

# Rutas a las carpetas de entrada y salida
folder_paths = [
    '/content/gdrive/My Drive/A___Doctorado/datos_thesis/Catulo',
    '/content/gdrive/My Drive/A___Doctorado/datos_thesis/Tibulo',
    '/content/gdrive/My Drive/A___Doctorado/datos_thesis/Propercio',
]
output_folder = Path('/content/gdrive/My Drive/A___Doctorado/datos_thesis/Output_TEI')

# Detalles de la edición original para el encabezado TEI
editions = {
    'Catulo': ("Catullus; edited by Elmer Truesdell Merrill", "1893", "http://archive.org/details/catulluseditedby00catuuoft"),
    'Tibulo': ("Tibulli aliorumque carminum libri tres. Scriptorum classicorum bibliotheca Oxoniensis", "1915", "https://archive.org/details/tibullialiorumqu00tibuoft"),
    'Propercio': ("Sex. Propertii Elegiae", "1898", "https://archive.org/details/sexpropertiieleg00prop")
}
```

A few minutes editing tasks

The first tasks involved automatically generating **key document elements** like the **header** and **body**, along with counting and **automatically tagging verses** and **stanzas**.

```python
def create_header(poem_title, author, edition, year, url):
    header = Element('teiHeader')
    file_desc = SubElement(header, 'fileDesc')
    title_stmt = SubElement(file_desc, 'titleStmt')
    title = SubElement(title_stmt, 'title')
    title.text = poem_title
    author_el = SubElement(title_stmt, 'author')
    author_el.text = author

    publication_stmt = SubElement(file_desc, 'publicationStmt')
    publisher = SubElement(publication_stmt, 'publisher')
    publisher.text = "Digital Edition"

    source_desc = SubElement(file_desc, 'sourceDesc')
    bibl = SubElement(source_desc, 'bibl')
    bibl.text = f"{edition}, {year}. Available at: {url}"

    return header
```

A few minutes editing tasks

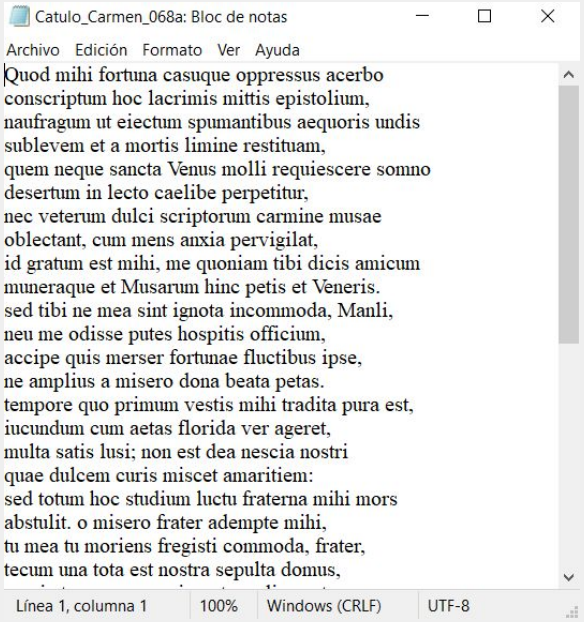
The first tasks involved automatically generating **key document elements** like the **header** and **body**, along with counting and **automatically tagging verses** and **stanzas**.

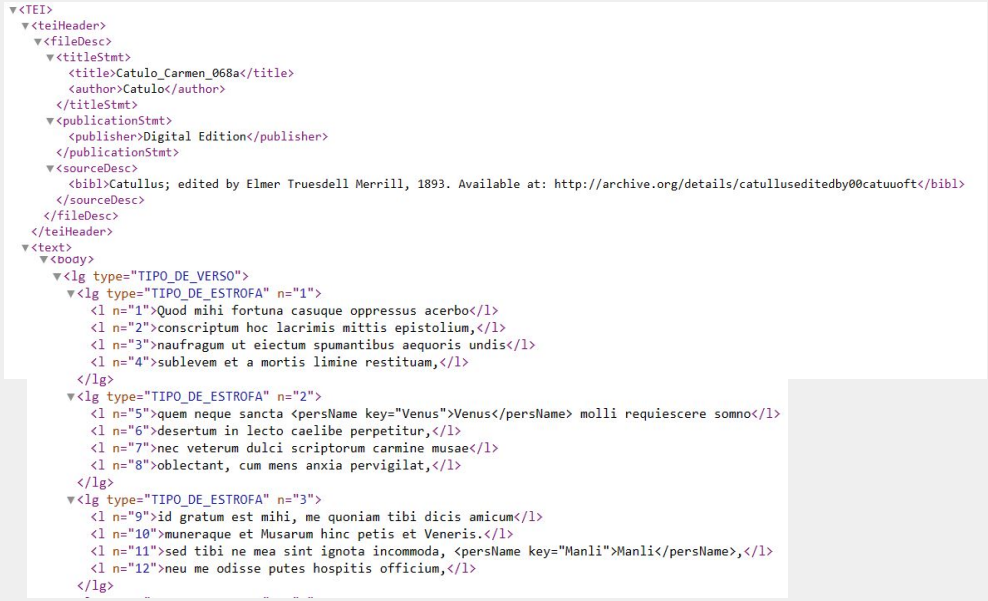
```python
from xml.etree import ElementTree
from pathlib import Path

# Función para contar versos en el texto y actualizar el archivo TEI
def count_verses_and_update_tei(tei_file_path):
    print(f"Procesando archivo: {tei_file_path}")
    # Cargar el archivo TEI existente
    tree = ElementTree.parse(tei_file_path)
    root = tree.getroot()
    body = root.find('.//body')
```

```python
# Contar versos y estrofas
verses = text.split('\n')  # Suponiendo que cada verso está separado por un salto de línea
stanza_count = 0
verse_count = 0
current_stanza = None

for verse in verses:
    if verse.strip():  # Asegurarse de que no es una línea vacía
        if current_stanza is None:
            current_stanza = ElementTree.SubElement(body, 'lg')
            stanza_count += 1
            print(f"Nueva estrofa creada. Total estrofas: {stanza_count}")
        verse_element = ElementTree.SubElement(current_stanza, 'l')
        verse_element.text = verse
        verse_count += 1
        print(f"Verso añadido: {verse}")
    else:
        current_stanza = None  # Resetear la estrofa si hay una línea en blanco
        print("Línea en blanco encontrada, estrofa reseteada.")

# Eliminar el párrafo original después de procesar su contenido
body.remove(paragraph)

# Agregar información del recuento al body
count_info = ElementTree.SubElement(body, 'count')
count_info.text = f'Total verses: {verse_count}, Total stanzas: {stanza_count}'
print(f"Total de versos: {verse_count}, Total de estrofas: {stanza_count}")

# Guardar los cambios en el archivo
tree.write(tei_file_path, encoding='unicode', xml_declaration=True)
```

A few minutes editing tasks

The first tasks involved automatically generating **key document elements** like the **header** and **body**, along with counting and **automatically tagging verses** and **stanzas**.





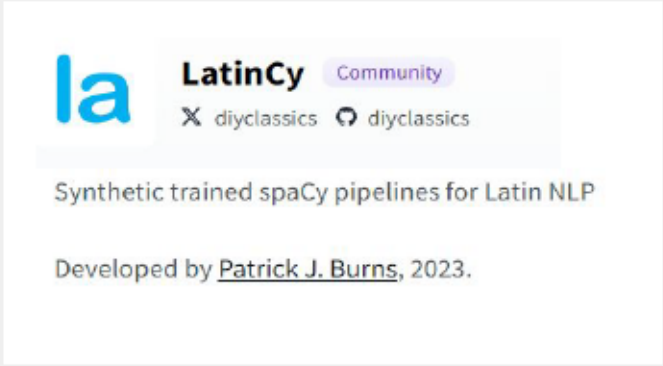
TEI 2024

2th Task: Automatic Extraction and Tagging of Entities

Subsequently, using the advanced **spaCy model** for Latin, **LatinCy**, the names of people and places were automatically extracted and **tagged** with the corresponding labels.

```
!pip install spacy
```

```
!pip install https://huggingface.co/latincy/la_core_web_lg/resolve/main/la_core_web_lg-any-py3-none-any.whl
```



2th Task: Automatic Extraction and Tagging of Entities

Subsequently, using the advanced **spaCy model** for Latin, **LatinCy**, the names of people and places were automatically extracted and **tagged** with the corresponding labels.

```python
import spacy
from pathlib import Path

# Cargar el modelo de spaCy
nlp = spacy.load("la_core_web_lg")

# Rutas a las carpetas de entrada
folder_paths = [
    '/content/gdrive/My Drive/A___Doctorado/datos_thesis/Catulo',
    '/content/gdrive/My Drive/A___Doctorado/datos_thesis/Tibulo',
    '/content/gdrive/My Drive/A___Doctorado/datos_thesis/Propercio',
]

# Listas para almacenar entidades
person_entities = []
location_entities = []
```

TEI 2024

Automatic Extraction and Tagging of Entities

```python
# Procesar cada archivo en las carpetas
for folder_path in folder_paths:
    folder = Path(folder_path)
    print(f"Procesando archivos en {folder.name}:")
    for file_path in folder.glob('*.txt'):
        with file_path.open('r', encoding='utf-8') as file:
            text = file.read()
            doc = nlp(text)
            print(f"Entidades en {file_path.name}:")
            for ent in doc.ents:
                if ent.label_ == "PERSON":
                    person_entities.append(ent.text)
                elif ent.label_ == "LOC":
                    location_entities.append(ent.text)
                print(f"{ent.text} ({ent.label_})")
            print("-" * 30)

# Guardar las entidades en archivos
with open('/content/gdrive/My Drive/A___Doctorado/datos_thesis/person_entities.txt', 'w', encoding='utf-8') as f:
    for person in person_entities:
        f.write(person + '\n')

with open('/content/gdrive/My Drive/A___Doctorado/datos_thesis/location_entities.txt', 'w', encoding='utf-8') as f:
    for location in location_entities:
        f.write(location + '\n')
```

Automatic Extraction and Tagging of Entities

Subsequently, using the advanced spaCy model for Latin, LatinCy, names of people and places were automatically extracted and tagged with the corresponding labels.

```
Procesando archivos en Catulo:
Entidades en Catulo_Carmen_029.txt:
Mamurram (PERSON)
Britannia (LOC)
Cinaede (PERSON)
Romule (PERSON)
Adoneus (PERSON)
Romule (PERSON)
Tagus (LOC)
Galliae (LOC)
Britanniae (LOC)
------------------------------
Entidades en Catulo_Carmen_014b.txt:
------------------------------
Entidades en Catulo_Carmen_012.txt:
Marrucine (PERSON)
Asini (PERSON)
Pollioni (PERSON)
Saetaba (PERSON)
Fabullus (PERSON)
Veranius (PERSON)
Veraniolum (PERSON)
Fabullum (PERSON)
```

```
Procesando archivos en Catulo:
Entidades en Catulo_Carmen_029.txt:
Mamurram (PERSON)
Britannia (LOC)
Cinaede (PERSON)
Romule (PERSON)
Adoneus (PERSON)
Romule (PERSON)
Tagus (LOC)
Galliae (LOC)
Britanniae (LOC)
```

```
Entidades en Propercio_Liber_3_Poema_20.txt:
Africa (LOC)
Luna (LOC)
Venus (PERSON)
Amor (PERSON)
```

```
Entidades en Propercio_Liber_1_Poema_13.txt:
Galle (LOC)
Galle (LOC)
Galle (LOC)
Enipeo (PERSON)
Taenarius (PERSON)
Iove (PERSON)
Ledae (PERSON)
Ledae (PERSON)
```

Automatic Extraction and Tagging of Entities

Subsequently, using the advanced spaCy model for Latin, LatinCy, names of people and places were automatically extracted and tagged with the corresponding labels.

```
Procesando archivos en Catulo:
Entidades en Catulo_Carmen_029.txt:
Mamurram (PERSON)
Britannia (LOC)
Cinaede (PERSON)
Romule (PERSON)
Adoneus (PERSON)
Romule (PERSON)
Tagus (LOC)
Galliae (LOC)
Britanniae (LOC)
------------------------------
Entidades en Catulo_Carmen_014b.txt:
------------------------------
Entidades en Catulo_Carmen_012.txt:
Marrucine (PERSON)
Asini (PERSON)
Pollioni (PERSON)
Saetaba (PERSON)
Fabullus (PERSON)
Veranius (PERSON)
Veraniolum (PERSON)
Fabullum (PERSON)
```

```
Procesando archivos en Catulo:
Entidades en Catulo_Carmen_029.txt:
Mamurram (PERSON)
Britannia (LOC)
Cinaede (PERSON)
Romule (PERSON)
Adoneus (PERSON)
Romule (PERSON)
Tagus (LOC)
Galliae (LOC)
Britanniae (LOC)
```

```
Entidades en Propercio_Liber_3_Poema_20.txt:
Africa (LOC)
Luna (LOC)
Venus (PERSON)
Amor (PERSON)
```

```
Entidades en Propercio_Liber_1_Poema_13.txt:
Galle (LOC)
Galle (LOC)
Galle (LOC)
Enipeo (PERSON)
Taenarius (PERSON)
Iove (PERSON)
Ledae (PERSON)
Ledae (PERSON)
```

It was perfect until... The Metaphorical Use Paradox

What happens when gods are used **metaphorically** to represent places? For example, Uranus for heaven or Tethys and Oceanus for the sea or ocean? There is our first editorial decision to make...

Thetis (PERSON)
Nereine (PERSON)
Oceanus (PERSON)

Uraniae (LOC)

Automatic Extraction and Tagging of Entities

```python
import spacy
from pathlib import Path
from lxml import etree

# Cargar el modelo de spaCy
nlp = spacy.load("la_core_web_lg")

# Función para crear el elemento TEI correcto
def create_tei_element(entity):
    if entity.label_ == "PERSON":
        return f'<persName key="{entity.text}">{entity.text}</persName>' #SACAR ATRIBUTO KEY
    elif entity.label_ == "LOC":
        return f'<placeName key="{entity.text}">{entity.text}</placeName>'
    return entity.text

# Rutas a las carpetas de entrada
folder_paths = [
    '/content/gdrive/My Drive/A___Doctorado/datos_thesis/Output_TEI'
]

```

Automatic Extraction and Tagging of Entities

```
# Procesar cada archivo TEI en las carpetas
for folder_path in folder_paths:
    folder = Path(folder_path)
    print(f"Procesando archivos en {folder.name}:")
    for file_path in folder.glob('*.xml'):  # Controlar la extensión correcta para los archivos TEI
        tree = etree.parse(str(file_path))
        root = tree.getroot()
        # Procesar cada línea de poesía dentro de los elementos <l>
        for l in root.xpath('//text//body//lg//l'):
            original_text = l.text
            doc = nlp(original_text)
            new_text = original_text
            for ent in reversed(doc.ents):  # Reversed para no alterar los índices mientras reemplazamos
                start = ent.start_char
                end = ent.end_char
                tei_tag = create_tei_element(ent)
                new_text = new_text[:start] + tei_tag + new_text[end:]
                print(f"Reemplazando '{ent.text}' por '{tei_tag}' en línea {l.get('n')}")

            # Actualizar el texto de la línea
            l.text = new_text
```

Automatic Extraction and Tagging of Entities

```
Procesando archivos en Output_TEI:
Reemplazando 'Uraniae' por '<placeName key="Uraniae">Uraniae</placeName>' en línea 2
Reemplazando 'Manilo' por '<persName key="Manilo">Manilo</persName>' en línea 16
Reemplazando 'Vinia' por '<persName key="Vinia">Vinia</persName>' en línea 16
Reemplazando 'Phrygium' por '<placeName key="Phrygium">Phrygium</placeName>' en línea 18
Reemplazando 'Asia' por '<placeName key="Asia">Asia</placeName>' en línea 22
Reemplazando 'Thespiae' por '<placeName key="Thespiae">Thespiae</placeName>' en línea 27
Reemplazando 'Aonios' por '<persName key="Aonios">Aonios</persName>' en línea 28
Reemplazando 'Aganippe' por '<persName key="Aganippe">Aganippe</persName>' en línea 30
Reemplazando 'Venus' por '<persName key="Venus">Venus</persName>' en línea 61
Reemplazando 'Au-' por '<persName key="Au-">Au-</persName>' en línea 83
Reemplazando 'Oceano' por '<placeName key="Oceano">Oceano</placeName>' en línea 86
Reemplazando 'Talasio' por '<persName key="Talasio">Talasio</persName>' en línea 129
Reemplazando 'Tyrio' por '<placeName key="Tyrio">Tyrio</placeName>' en línea 167
Reemplazando 'Venus' por '<persName key="Venus">Venus</persName>' en línea 193
Reemplazando 'Venus' por '<persName key="Venus">Venus</persName>' en línea 197
Reemplazando 'Africi' por '<placeName key="Africi">Africi</placeName>' en línea 201
Reemplazando 'Torquatus' por '<persName key="Torquatus">Torquatus</persName>' en línea 211
Reemplazando 'Manlio' por '<persName key="Manlio">Manlio</persName>' en línea 217
Reemplazando 'Telemacho' por '<persName key="Telemacho">Telemacho</persName>' en línea 224
Archivo actualizado: Catulo_Catulo_Carmen_061_TEI.xml
Reemplazando 'Libystinis' por '<placeName key="Libystinis">Libystinis</placeName>' en línea 1
Reemplazando 'Scylla' por '<persName key="Scylla">Scylla</persName>' en línea 2
Archivo actualizado: Catulo_Catulo_Carmen_060_TEI.xml
Reemplazando 'Lesbia' por '<persName key="Lesbia">Lesbia</persName>' en línea 7
Reemplazando 'Catulle' por '<persName key="Catulle">Catulle</persName>' en línea 13
Archivo actualizado: Catulo_Catulo_Carmen_051_TEI.xml
```

What else can I do to enhance my edition?

What else can I do to enhance my edition?

- In addition, searches for standardized metadata were carried out using external **APIs**, and consulting databases such as Virtual International Authority File (VIAF), Pleiades project, and Wikidata.
- This allowed for the **retrieval** of **standardized identifiers**, **rich** and **curated information**, and **images** of historical places and characters.

VIAF

PLEIADES

WIKIDATA

APIs

- API stands for **Application Programming Interface**, which is a **set of rules and protocols** that allows different software applications to **communicate** with each other.
- APIs provide a way for developers to access data or services from external systems without needing to know the internal workings of those systems.

VIAF

WIKIDATA

What is VIAF?

- VIAF, or **Virtual International Authority File**, is a system that links **authority records of names** (people, organizations) from national libraries and other institutions worldwide.
- It helps standardize name **identifiers** across **databases** and **provides unique, persistent identifiers for individuals**, making it easier to connect various information sources.

VIAF

VIAF API

- We **searched** for the VIAF (Virtual International Authority File) **IDs** of a **list of person names**.
- We used the VIAF API to find the matching records, and then saved the found results (including the name, VIAF ID, and link) into one file and the names that were not found into another.


```python
import requests

def get_viaf_details(name):
    url = f"http://viaf.org/viaf/AutoSuggest?query={name}"
    try:
        response = requests.get(url)
        data = response.json()
        if 'result' in data and data['result']:
            result = {
                "term": data['result'][0]['term'],
                "viaf_id": data['result'][0]['viafid'],
                "link": f"https://viaf.org/viaf/{data['result'][0]['viafid']}"
            }
            print(f"Nombre buscado: {name}, Nombre encontrado: {result['term']}, {result['viaf_id']}, {result['link']}")
            return result
        else:
            print(f"Nombre buscado: {name}, Not Found")
            return None
    except requests.RequestException as e:
        print(f"Error en la solicitud para {name}: {e}")
        return None

# Cargar la lista de nombres de personas desde un archivo y eliminar duplicados
with open('/content/gdrive/My Drive/A__Doctorado/datos_thesis/person_entities.txt', 'r', encoding='utf-8') as f:
    person_entities = set(line.strip() for line in f if line.strip())
```

VIAF API

The code worked perfectly, but the **volume of data retrieved was enormous!!!!**

```
Nombre buscado: Amabo, Nombre encontrado: Amabokoboko (Rugby team), 159851024, https://viaf.org/viaf/159851024
Nombre buscado: Thamyrae, Not Found
Nombre buscado: Tarquinii, Nombre encontrado: Tarquinii, 291167802705917770162, https://viaf.org/viaf/291167802705917770162
Nombre buscado: Licini, Nombre encontrado: Licinio Refice, 1883-1954, 14959385, https://viaf.org/viaf/14959385
Nombre buscado: Acmen, Nombre encontrado: Acmenide, 9514159477862627990005, https://viaf.org/viaf/9514159477862627990005
Nombre buscado: Claudius, Nombre encontrado: Claudius Galenus, ca. 129-ca. 210/216, 44299175, https://viaf.org/viaf/44299175
Nombre buscado: Rhenus, Nombre encontrado: Rhenus, 1575-1642, 49288294, https://viaf.org/viaf/49288294
Nombre buscado: Risi, Nombre encontrado: Risi, Dino, 1916-2008, 19761937, https://viaf.org/viaf/19761937
Nombre buscado: Crannonis, Not Found
Nombre buscado: Marathum, Not Found
Nombre buscado: Artaciis, Not Found
Nombre buscado: Threicia, Not Found
Nombre buscado: Romule, Nombre encontrado: Romuleus, Bartholomaeus, -1588, 38006675, https://viaf.org/viaf/38006675
Nombre buscado: Nereine, Not Found
Nombre buscado: Danai, Nombre encontrado: Danai Gurira, 1978-, 274583701, https://viaf.org/viaf/274583701
Nombre buscado: Tappone, Nombre encontrado: Tapponen, Sakari, 1538163706790929421526, https://viaf.org/viaf/1538163706790929421526
```

Pleiades Project

- Is an open, collaborative resource providing **detailed information** about **ancient places**, primarily from the Greco-Roman world.
- It links ancient and modern geography using **digital mapping tools** and **open data** like **geographic information systems (GIS)**, allowing users to visualize the locations on a map and see how ancient geography relates to today's world.

TEI 2024

Data Download from Pleiades

First, the data was downloaded from the Pleiades Downloads page, where various datasets are available in formats like CSV, GeoJSON, and KML.
The chosen datasets were:

- Names CSV: Contains information on attested names of ancient places, their romanized forms, and links to Pleiades records.
- Locations CSV: Includes geographical coordinates (points or polygons) of registered places.

Processing Data from Pleiades

- We **filtered entities** that match the predefined list of historical names.
- Checked for matches in both the **attested** and **romanized** forms.
- The filtered data was merged with **geographical information** (**points** and **polygons**). Finally, the results are saved in a new CSV file and displayed.

```python
import pandas as pd

# Cargar los datos desde los archivos CSV
df_names = pd.read_csv('/content/gdrive/My Drive/A___Doctorado/datos_thesis/Pleiades_CSVs/names.csv', usecols=[
    'place_id', 'attested_form', 'romanized_form_1', 'romanized_form_2', 'romanized_form_3', 'uri'
])
df_location_points = pd.read_csv('/content/gdrive/My Drive/A___Doctorado/datos_thesis/Pleiades_CSVs/location_points.csv', usecols=[
    'place_id', 'geometry_wkt'
])
df_location_polygons = pd.read_csv('/content/gdrive/My Drive/A___Doctorado/datos_thesis/Pleiades_CSVs/location_polygons.csv', usecols=[
    'place_id', 'geometry_wkt'
])

# Filtro para encontrar las entidades nombradas
entidades = ['Castalia', 'Scythiae', 'Veios', 'Iasonia', 'Assyrios', 'Rhamnusia', 'Lethaeam', 'Romae', 'Spartana', 'Castra', 'Asis', 'Creta', 'Tyrrhenos', 'Calliope', 'Thraciam', '
```

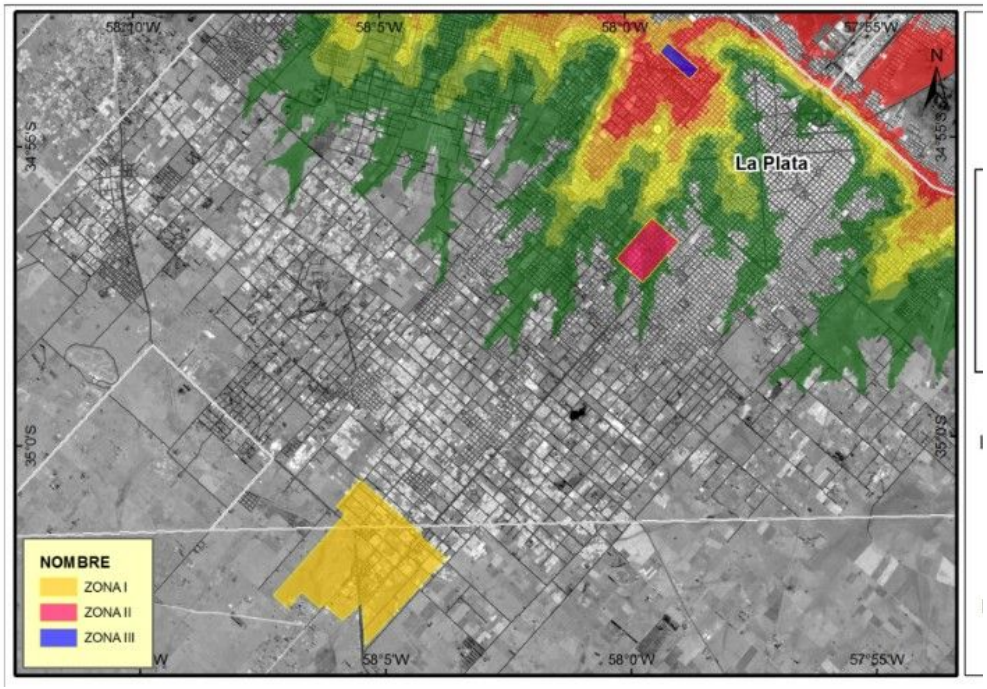
Processing Data from Pleiades

	uri	place_id	attested_form	\
0	https://pleiades.stoa.org/places/993/gallia	993	NaN	
1	https://pleiades.stoa.org/places/991401/august...	991401	NaN	
2	https://pleiades.stoa.org/places/991409/august...	991409	NaN	
3	https://pleiades.stoa.org/places/991373/creta	991373	NaN	
4	https://pleiades.stoa.org/places/991386/phrygia-i	991386	NaN	

	romanized_form_1	romanized_form_2	romanized_form_3	entidad_buscada	\
0	Gallia	NaN	NaN	Gallia	
1	Augusta Euphratensis	NaN	NaN	Euphraten	
2	Augusta Libanensis	NaN	NaN	Liba	
3	Creta	NaN	NaN	Creta	
4	Phrygia I	NaN	NaN	Phrygia	

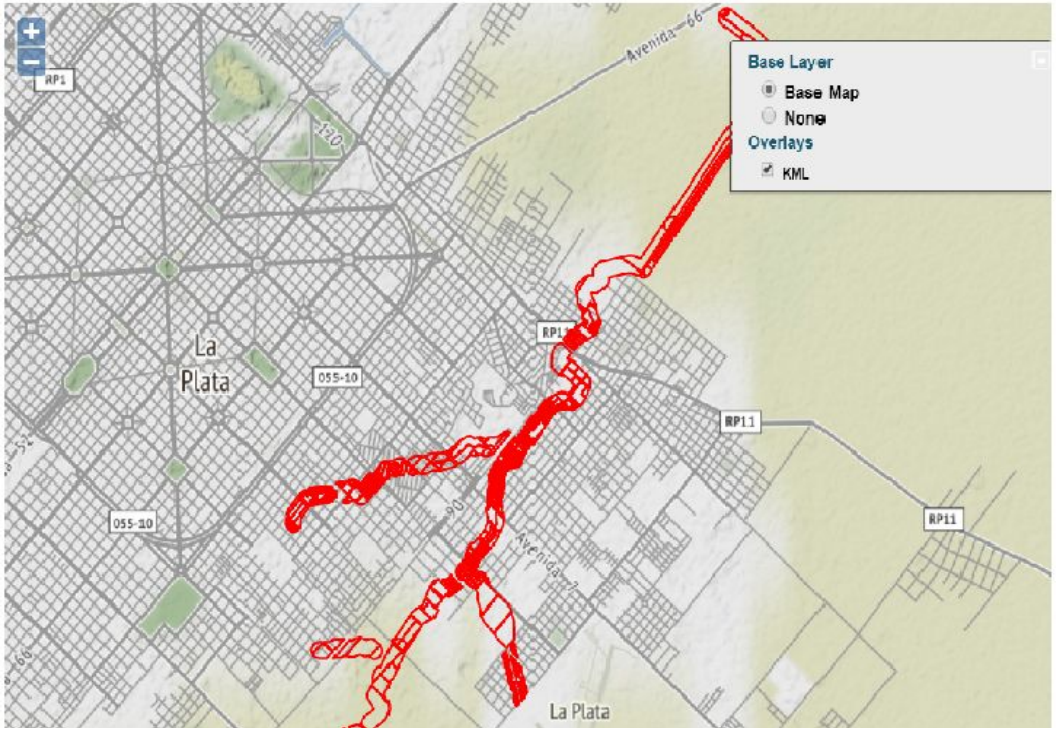
	geometry_wkt	geometry_wkt_polygon
0	NaN	NaN
1	NaN	POLYGON ((35 35, 40 35, 40 40, 35 40, 35 35))
2	NaN	POLYGON ((35 30, 40 30, 40 35, 35 35, 35 30))
3	NaN	POLYGON ((20 35, 25 35, 25 40, 20 40, 20 35))
4	NaN	POLYGON ((25 35, 30 35, 30 40, 25 40, 25 35))

Processing Data from Pleiades



Processing Data from Pleiades

Processing Data from Pleiades



Wikidata API

- Wikidata is a free, collaborative, multilingual database that serves as a **central repository for structured data**.
- Wikidata offers open, structured data that can be linked to various projects, making it ideal for research, data analysis, and automation.
- It connects data from other Wikimedia projects (Wikipedia, Wikimedia Commons) and external databases, serving as a hub for cross-referencing.
- Wikidata **provides unique identifiers** (Q numbers) for **people**, **places**, **objects**, and **concepts**, allowing easy access and integration across datasets.

Wikidata API


```python
import requests

def search_wikidata(place_name):
    url = "https://www.wikidata.org/w/api.php"
    params = {
        "action": "wbsearchentities",
        "language": "en",
        "format": "json",
        "search": place_name,
        "type": "item"
    }
    try:
        response = requests.get(url, params=params)
        data = response.json()
        results = []
        if data['search']:
            for result in data['search']:
                details = get_entity_images(result['id'])
                results.append({
                    "id": result['id'],
                    "label": result['label'],
                    "images": details
                })
            return results
        else:
            return []
    except requests.RequestException as e:
        return {"error": f"Request failed: {e}"}

```

Wikidata API


```python
def get_entity_images(entity_id):
    url = "https://www.wikidata.org/w/api.php"
    params = {
        "action": "wbgetclaims",
        "entity": entity_id,
        "property": "P18",
        "format": "json"
    }
    try:
        response = requests.get(url, params=params)
        data = response.json()
        images = []
        if 'claims' in data and 'P18' in data['claims']:
            for claim in data['claims']['P18'][:10]:  # Limit to 10 images
                image_title = claim['mainsnak']['datavalue']['value']
                image_url = f"https://commons.wikimedia.org/wiki/File:{image_title.replace(' ', '_')}"
                images.append(image_url)
        return images
    except requests.RequestException:
        return []
```

Wikidata API

```
Results for Scythiae:
Results for Veios:
  - Veii (Q677935):
      https://commons.wikimedia.org/wiki/File:Colonne_di_Veio_a_palazzo_Wedekind.JPG
      https://commons.wikimedia.org/wiki/File:Veii.png
Results for Iasonia:
  - Jasonia (Pontus) (Q17635213):
Results for Assyrios:
Results for Rhamnusia:
  - Rhamnusia Virgo (Q59686630):
  - Nemesis (Q185747):
      https://commons.wikimedia.org/wiki/File:Statue_Nemesis_Louvre_Ma4873.jpg
Results for Lethaeam:
Results for Romae:
  - Parilia (Q666621):
      https://commons.wikimedia.org/wiki/File:Suvée_Festa_di_Pales.jpg
  - Romance Is a Bonus Book (Q60174208):
  - Romaeuropa Festival (Q55829521):
  - Romaero (Q387165):
      https://commons.wikimedia.org/wiki/File:Baneasa_Airport_4.jpg
  - Romaen Campbell (Q116476699):
  - Romani people (Q8060):
      https://commons.wikimedia.org/wiki/File:Bain_rituel_dans_la_mer_aux_Saintes-Maries.jpg
      https://commons.wikimedia.org/wiki/File:Romanlar_(Istanbul).JPG
  - Romeo and Juliet (Q83186):
      https://commons.wikimedia.org/wiki/File:Romeo_and_juliet_brown.jpg
      https://commons.wikimedia.org/wiki/File:Romeo_Juliet.jpg
      https://commons.wikimedia.org/wiki/File:DickseeRomeoandJuliet.jpg
      https://commons.wikimedia.org/wiki/File:Rigaud-RomeoJuliet.jpg
Results for Spartana:
```

Wikidata API

```
Results for Scythiae:
Results for Veios:
  - Veii (Q677935):
      https://commons.wikimedia.org/wiki/File:Colonne_di_Veio_a_palazzo_Wedekind.JPG
      https://commons.wikimedia.org/wiki/File:Veii.png
Results for Iasonia:
  - Jasonia (Pontus) (Q17635213):
Results for Assyrios:
Results for Rhamnusia:
  - Rhamnusia Virgo (Q59686630):
  - Nemesis (Q185747):
      https://commons.wikimedia.org/wiki/File:Statue_Nemesis_Louvre_Ma4873.jpg
Results for Lethaeam:
Results for Romae:
  - Parilia (Q666621):
      https://commons.wikimedia.org/wiki/File:Suvée_Festa_di_Pales.jpg
  - Romance Is a Bonus Book (Q60174208):
  - Romaeuropa Festival (Q55829521):
  - Romaero (Q387165):
      https://commons.wikimedia.org/wiki/File:Baneasa_Airport_4.jpg
  - Romaen Campbell (Q116476699):
  - Romani people (Q8060):
      https://commons.wikimedia.org/wiki/File:Bain_rituel_dans_la_mer_aux_Saintes-Maries.jpg
      https://commons.wikimedia.org/wiki/File:Romanlar_(Istanbul).JPG
  - Romeo and Juliet (Q83186):
      https://commons.wikimedia.org/wiki/File:Romeo_and_juliet_brown.jpg
      https://commons.wikimedia.org/wiki/File:Romeo_Juliet.jpg
      https://commons.wikimedia.org/wiki/File:DickseeRomeoandJuliet.jpg
      https://commons.wikimedia.org/wiki/File:Rigaud-RomeoJuliet.jpg
Results for Spartana:
```

TEI 2024

Wikidata API

File:Statue Nemesis Louvre Ma4873.jpg

From Wikimedia Commons, the free media repository

Download Use this file Use this file Email a link Information

The moral of this experiment

- Some tasks, like the automatic generation of TEI files and automatic tagging, work very well and streamline the editing process.
- However, they still require a **final review by the editor** to ensure accuracy and quality control, making the human touch essential for the last stage of refinement.

- The three prototypes for improving the digital edition were successful, as **we successfully** integrated **APIs** from **VIAF**, **Pleiades**, and **Wikidata**.
- However, we have become victims of our own success. Now, we face **a new bottleneck**: processing **the massive amount of data we've retrieved**.
- With thousands of names, places, and images, **the challenge** is no longer in retrieving or manually searching for data but in **efficiently processing and filtering the correct information locally**.
- Managing this overwhelming volume of data requires new strategies and tools for optimization.

UNIVERSIDAD NACIONAL DE LA PLATA

TEI 2024

TEI Text Encoding Initiative

USAL UNIVERSIDAD DEL SALVADOR

Carlos J. Nusch
carlosnusch@prebi.unlp.edu.ar

PREBI SEDICI
prebi.unlp.edu.ar sedici.unlp.edu.ar