CHVorl4

November 26, 2016

1 Vorlesung 4: Strukturen von Texten 2

Reguläre Ausdrücke: http://www.regexe.de/hilfe.jsp https://pymotw.com/2/re

Pandas: http://www.data-analysis-in-python.org/3_pandas.html : https://bitbucket.org/hrojas/learn-pandas

```
In [1]: import json
        import pandas as pd
        import re
        import numpy as np
```

1.1 Text als Dataframe *Poleis*

```
In [2]: # json.load erzeugt ein dictionary der JSON Daten
        with open('chapter1.json') as json_data:
            PoleisRawData = json.load(json_data)
```

```
In [3]: # list() erstellt eine Liste der keys
        PoleisKeyList = list(PoleisRawData.keys())
        PoleisKeyList
```

```
Out[3]: ['18. Heloron ',
         '41. Naxos ',
         '11. Alaisa ',
         '34. Lipara ',
         '23. Herbita ',
         '48. Tauromenion ',
         '25. Hippana ',
         '21. Herakleia 2 ',
         '29. Kasmenai ',
         '30. Katane ',
         '27. Kallipolis ',
         '42. Petra ',
         '37. Morgantina ',
         '9. Akragas ',
         '44. Selinous ',
         '24. Himera ',
         '13. Apollonia ',
         '17. Gela ',
         '8. Aitna ',
         '36. Megara ',
         '35. *Longane ',
         '33. Leontinoi ',
```

```
 '38. Mylai ',
 '49. Tyndaris ',
 '39. Mytistratos ',
 '12. Alontion ',
 '32. Kephaloidion ',
 '40. Nakone ',
 '26 *Imachara ',
 '19. Henna ',
 '14. Engyon ',
 '16. Galeria ',
 '47. Syrakousai ',
 '5. Abakainon ',
 '50. (Tyrrhenoi)',
 '22. Herbes(s)os ',
 '20. Herakleia 1',
 '6. Adranon ',
 '15. Euboia ',
 '31. Kentoripa ',
 '43. Piakos ',
 '7. Agyrion ',
 '10. Akrai ',
 '45. (Sileraioi)',
 '28. Kamarina ',
 '51. Zankle ',
 '46. (Stielanaioi)']
```

In [4]:
```
#%%tutor --lang python3
for i in PoleisKeyList:
    if "Megara" in PoleisRawData[i]:
        print(i)
```

```
18. Heloron
44. Selinous
36. Megara
33. Leontinoi
47. Syrakousai
15. Euboia
46. (Stielanaioi)
```

In [5]:
```
# Liest das Dictionary als Dataframe ein. Namen der Poleis werden als Index benutzt
dfPoleis = pd.DataFrame([PoleisRawData]).transpose()
dfPoleis = dfPoleis.rename(columns={0: 'Beschreibung'})
dfPoleis.head()
```

Out[5]:

	Beschreibung
10. Akrai	(Akraios) Map 47. Lat. 37.05,long. 14.55. ...
11. Alaisa	(Alaisinos) Map 47. Lat. 38.00,long. 14.15...
12. Alontion	(Alontinos) Map 47. Lat. 38.05,long. 14.40...
13. Apollonia	(Apolloniates) Map 47. Lat. 38.00,long. 14.3...
14. Engyon	(Engyinos) Map 47. Lat. 37.45,long. 14.35...

2 Konstruktion neuer Merkmale

2.1 Textmuster mit regulären Ausdrücken

http://www.regexe.de/hilfe.jsp https://www.cheatography.com/davechild/cheat-sheets/regular-expressions/
http://www.coli.uni-saarland.de/courses/python1-10/folien/PythonI10-07.pdf

```
In [6]: # Konstruktion einer Liste mit sogenannten List-Comprehensions
        ListCities = [x[4:] for x in dfPoleis.index]
        ListCities
```

```
Out[6]: ['Akrai ',
         'Alaisa ',
         'Alontion ',
         'Apollonia ',
         'Engyon ',
         'Euboia ',
         'Galeria ',
         'Gela ',
         'Heloron ',
         'Henna ',
         'Herakleia 1',
         'Herakleia 2 ',
         'Herbes(s)os ',
         'Herbita ',
         'Himera ',
         'Hippana ',
         'Imachara ',
         'Kallipolis ',
         'Kamarina ',
         'Kasmenai ',
         'Katane ',
         'Kentoripa ',
         'Kephaloidion ',
         'Leontinoi ',
         'Lipara ',
         '*Longane ',
         'Megara ',
         'Morgantina ',
         'Mylai ',
         'Mytistratos ',
         'Nakone ',
         'Naxos ',
         'Petra ',
         'Piakos ',
         'Selinous ',
         '(Sileraioi)',
         '(Stielanaioi)',
         'Syrakousai ',
         'Tauromenion ',
         'Tyndaris ',
         'bakainon ',
         '(Tyrrhenoi)',
         'Zankle ',
```

```
 'dranon ',
 'gyrion ',
 'itna ',
 'kragas ']
```

In [7]:
```
# Extrahiere Name der Polis aus Index
dfPoleis['city'] = [x[4:] for x in dfPoleis.index]
dfPoleis.head()
```

Out[7]:

	Beschreibung	city
10. Akrai	(Akraios) Map 47. Lat. 37.05,long. 14.55. ...	Akrai
11. Alaisa	(Alaisinos) Map 47. Lat. 38.00,long. 14.15...	Alaisa
12. Alontion	(Alontinos) Map 47. Lat. 38.05,long. 14.40...	Alontion
13. Apollonia	(Apolloniates) Map 47. Lat. 38.00,long. 14.3...	Apollonia
14. Engyon	(Engyinos) Map 47. Lat. 37.45,long. 14.35...	Engyon

In [9]:
```
[re.findall("\d{1,2}",x) for x in dfPoleis.index]
```

Out[9]:
```
[['10'],
 ['11'],
 ['12'],
 ['13'],
 ['14'],
 ['15'],
 ['16'],
 ['17'],
 ['18'],
 ['19'],
 ['20', '1'],
 ['21', '2'],
 ['22'],
 ['23'],
 ['24'],
 ['25'],
 ['26'],
 ['27'],
 ['28'],
 ['29'],
 ['30'],
 ['31'],
 ['32'],
 ['33'],
 ['34'],
 ['35'],
 ['36'],
 ['37'],
 ['38'],
 ['39'],
 ['40'],
 ['41'],
 ['42'],
 ['43'],
 ['44'],
 ['45'],
 ['46'],
```

```
['47'],
['48'],
['49'],
['5'],
['50'],
['51'],
['6'],
['7'],
['8'],
['9']]
```

```
In [10]: # Extrahiere Nummer des Polis Eintrags
         dfPoleis['city_index'] = [int(re.findall('\d{1,2}', x)[0]) for x in dfPoleis.index
         dfPoleis.head()
```

Out[10]:

	Beschreibung	city	city_index
10. Akrai	(Akraios) Map 47. Lat. 37.05,long. 14.55. ...	Akrai	10
11. Alaisa	(Alaisinos) Map 47. Lat. 38.00,long. 14.15...	Alaisa	11
12. Alontion	(Alontinos) Map 47. Lat. 38.05,long. 14.40...	Alontion	12
13. Apollonia	(Apolloniates) Map 47. Lat. 38.00,long. 14.3...	Apollonia	13
14. Engyon	(Engyinos) Map 47. Lat. 37.45,long. 14.35...	Engyon	14

```
In [11]: # Sortiere die Zeilen nach der Spalte
         dfPoleis = dfPoleis.sort_values(by='city_index')
         dfPoleis.head(4)
```

Out[11]:

	Beschreibung	city	city_index
5. Abakainon	(Abakaininos) Map 47. Lat. 38.05,long. 15.05...	bakainon	5
6. Adranon	(Adranites) Map 47. Lat. 37.40,long. 14.50...	dranon	6
7. Agyrion	(Agyrinaios) Map 47. Lat. 37.40,long. 14.30...	gyrion	7
8. Aitna	(Aitnaios) Map 47.Location of Aitna I as ...	itna	8

2.2 Textmustersuche in der Beschreibung einer Polis

2.2.1 Neue Funktionen

```
In [12]: nl=dfPoleis["city"]
         nl.head()
```

```
Out[12]: 5. Abakainon     bakainon
         6. Adranon         dranon
         7. Agyrion         gyrion
         8. Aitna             itna
```

```
         9. Akragas         kragas
         Name: city, dtype: object

In [13]: lcity=list(dfPoleis["city"])
         lcity[0:3]

Out[13]: ['bakainon ', 'dranon ', 'gyrion ']

In [14]: for i in lcity:
             if re.search('oi',i):
                 print(i)

Euboia
Kephaloidion
Leontinoi
(Sileraioi)
(Stielanaioi)
(Tyrrhenoi)


In [15]: dfPoleis[dfPoleis["city"].str.contains("ag")]

Out[15]:                                                    Beschreibung     city  \
         9. Akragas   (Akragantinos) Map  47.  Lat. 37.20,long.  13...  kragas

                     city_index
         9. Akragas            9

In [16]: dfNeu=dfPoleis[dfPoleis["city"].str.contains("ag")]
```

2.2.2 Geographische Koordinaten

```
In [17]: def ListePattern(string,pattern):
             x = re.findall(pattern,string)
             if x:
                 return(x)
```

- (?<=Lat.) Group (?…) Passive (non-capturing) group
- ?<= Lookbehind assertion
- Lat.das string muster: “Lat.” mit “.” und " " als escape
- ?+.+ : space[optional wegen ?]digit[1 oder 2 wegen +].[escaped]digit[1 oder 2]

```
In [18]: ListePattern(dfPoleis["Beschreibung"][0],"(?<=Lat\.\s)\s?\d+\.\d+")

Out[18]: ['38.05']

In [19]: # gleiches auch für long.
         listLong=ListePattern(dfPoleis['Beschreibung'][0],"(?<=long\.\s)\s?\d+\.\d+")
         listLong

Out[19]: ['15.05']

In [20]: dfPoleis["Beschreibung"].apply(lambda row: ListePattern(row,"(?<=Lat\.\s)\s?\d+\.\

Out[20]: 5. Abakainon         [38.05]
         6. Adranon           [37.40]
         7. Agyrion           [37.40]
```

```
8. Aitna                 None
9. Akragas             [37.20]
10. Akrai              [37.05]
11. Alaisa             [38.00]
12. Alontion           [38.05]
13. Apollonia          [38.00]
14. Engyon             [37.45]
15. Euboia               None
16. Galeria              None
17. Gela               [37.05]
18. Heloron            [36.50]
19. Henna              [37.35]
20. Herakleia 1        [37.25]
21. Herakleia 2          None
22. Herbes(s)os          None
23. Herbita              None
24. Himera             [37.55]
25. Hippana            [37.40]
26 *Imachara             None
27. Kallipolis           None
28. Kamarina           [36.50]
29. Kasmenai           [37.05]
30. Katane             [37.30]
31. Kentoripa          [37.35]
32. Kephaloidion       [38.00]
33. Leontinoi          [37.15]
34. Lipara             [38.30]
35. *Longane           [38.05]
36. Megara             [37.10]
37. Morgantina         [37.25]
38. Mylai              [38.15]
39. Mytistratos        [37.35]
40. Nakone               None
41. Naxos              [37.50]
42. Petra                None
43. Piakos               None
44. Selinous           [37.35]
45. (Sileraioi)          None
46. (Stielanaioi)      [37.10]
47. Syrakousai         [37.05]
48. Tauromenion        [37.50]
49. Tyndaris           [38.10]
50. (Tyrrhenoi)          None
51. Zankle               None
Name: Beschreibung, dtype: object
```

In [21]:
```
dfPoleis['Latitude'] = dfPoleis['Beschreibung'].apply(lambda raw: ListePattern(raw

dfPoleis['Longitude'] = dfPoleis['Beschreibung'].apply(lambda raw: ListePattern(ra
```

In [22]:
```
dfPoleis.head(4)
```

Out[22]:

	Beschreibung	city	\
5. Abakainon	(Abakaininos) Map 47. Lat. 38.05,long. 15.05...	bakainon	
6. Adranon	(Adranites) Map 47. Lat. 37.40,long. 14.50...	dranon	

```
7. Agyrion     (Agyrinaios) Map  47.  Lat. 37.40,long.  14.30...    gyrion
8. Aitna       (Aitnaios) Map  47.Location  of  Aitna  I  as ...      itna

               city_index Latitude Longitude
5. Abakainon            5  [38.05]   [15.05]
6. Adranon              6  [37.40]  [ 14.50]
7. Agyrion              7  [37.40]  [ 14.30]
8. Aitna                8     None      None
```

2.2.3 Zitatnachweise, Namen, Jahreszahlen

```
In [23]: dfPoleis["Beschreibung"].iloc[0] # icloc is a method to refer to local index posit

Out[23]: '(Abakaininos) Map  47.  Lat. 38.05,long. 15.05.  Size  of  territory:  ?  Type:
```

2.3 Muster (Pattern) zur Erkennung der Literaturreferenzen

- Primärquellen

(Polyb. 1.18.2) (Diod. 13.85.4 (r 406)) (Diod. 13.108.2) (Hdt. 7.165; IGDS no. 182a) (Pind. Pyth. 6) (Thuc. 6.4.4: $\mu\mu$) (Xanthos (FGrHist 765) fr. 33; Arist. fr. 865)

- Sekundärquellen

(Karlsson (1995) 161 (Waele (1971) 195; Hinz (1998) 79)

- Jahreszahlen (dddd)

2.3.1 Testen der regulären Ausdrücke

Finde alle groß-geschriebenen Wörter mit mindestens 3 nachfolgenden kleinen Buchstaben.

```
In [24]: dfPoleis['Beschreibung'].apply(lambda raw: ListePattern(raw,'[A-Z][a-z]{3,10}'))[0

Out[24]: ['Abakaininos',
          'Size',
          'Type',
          'Diod',
          'Diod',
          'Steph',
          'Diod',
          'Steph',
          'Abakainon',
          'Diod',
          'Magon',
          'Agathokles',
          'Kamarina',
          'Leontinoi',
          'Katane',
          'Messana',
          'Diod',
          'Diod',
          'Dionysios',
          'Abakainon',
          'Tyndaris',
          'Diod',
```

```
 'Abakainon',
 'Tyndaris',
 'Tripi',
 'However',
 'Diodorus',
 'Carthaginia',
 'Manni',
 'Dionysios',
 'There',
 'Greek',
 'Greek',
 'Villard',
 'Leontinoi',
 'Bacci',
 'Spigo',
 'Greek',
 'Abakainon',
 'Zeus',
 'Apollo',
 'Demeter',
 'Persephone',
 'Head',
 'Bertino',
 'Sicily',
 'Probably',
 'Timoleon',
 'Head',
 'Bertino',
 'Sicily',
 'Dioskouros',
 'Tyndaris',
 'Italy',
 'Bertino']
```

Finde alle Ausdrücke wie oben, denen ein Punkt folgt, mit anschließenden Zifferfolgen der Form [Ziffern][Punkt][Ziffern][Punkt][Ziffern]

```
In [27]: dfPoleis['Beschreibung'].apply(lambda raw: ListePattern(raw,'[A-Z][a-z]{1,10}\. \(
```

```
Out[27]: ['Diod. 14.90.3',
          'Diod. 19.65.6',
          'Diod. 14.78.5',
          'Diod. 14.90.3',
          'Diod. 19.65.6',
          'Diod. 19.110.4']
```

Finde alle Ausdrücke wie oben, wobei statt des Punktes nach den kleinen Buchstaben auch zwei Leerzeichen und eine runde Klammer folgen können

2.4 Muster zur Erkennung von Namen

Empedokles (496) Theron (476) Timoleon c. 338

```
In [28]: dfPoleis['Beschreibung'].apply(lambda raw: ListePattern(raw,'[A-Z][a-z]{1,10}[\.|
```

```
Out[28]: ['Diod. 14.90.3',
          'Diod. 19.65.6',
          'Diod. 14.78.5',
          'Diod. 14.90.3',
          'Diod. 19.65.6',
          'Diod. 19.110.4',
          'Manni  ( 1976',
          'Villard  ( 1954)',
          'Spigo  ( 1997',
          'Bertino  ( 1975)',
          'Bertino  ( 1975)',
          'Bertino  ( 1975)']
```

```
In [29]: dfPoleis['Namen'] = dfPoleis['Beschreibung'].apply(lambda raw: ListePattern(raw,'
```

```
In [30]: dfPoleis['Quellen'] = dfPoleis['Beschreibung'].apply(lambda raw: ListePattern(raw,
```

```
In [31]: dfPoleis.head(4)
```

```
Out[31]:                                                    Beschreibung       city  \
         5. Abakainon   (Abakaininos) Map  47.  Lat. 38.05,long. 15.05...  bakainon
         6. Adranon     (Adranites) Map  47.  Lat. 37.40,long.  14.50...    dranon
         7. Agyrion     (Agyrinaios) Map  47.  Lat. 37.40,long.  14.30...    gyrion
         8. Aitna       (Aitnaios) Map  47.Location  of  Aitna  I  as ...      itna

                        city_index Latitude Longitude  \
         5. Abakainon            5  [38.05]   [15.05]
         6. Adranon              6  [37.40]  [ 14.50]
         7. Agyrion              7  [37.40]  [ 14.30]
         8. Aitna                8     None      None

                                                                  Namen  \
         5. Abakainon   [Abakaininos, Size, Type, Diod, Diod, Steph, D...
         6. Adranon     [Adranites, Size, Type, Diod, Steph, Diod, Adr...
         7. Agyrion     [Agyrinaios, Size, Type, Diod, Ptol, Geog, Ste...
         8. Aitna       [Aitnaios, Location, Aitna, Katane, Aitna, Dio...

                                                               Quellen
         5. Abakainon   [Diod. 14.90.3, Diod. 19.65.6, Diod. 14.78.5, ...
         6. Adranon     [Diod. 14.37.5, Diod. 16.68.9, Diod. 14.37.5, ...
         7. Agyrion     [Byz. 23.19), Diod. 16.82.4, Moggi  ( 1976), D...
         8. Aitna       [Diod. 11.49.1, Diod. 11.49.1, Diod. 11.66.4, ...
```

3 Datenvalidierung

3.1 Bewertung des Modells mit Performanz- (Konfusions-)matrix

Diskussion der Performanzmatrix: vier Fälle - soll match vs. tatsächlicher match - nicht soll match vs. tatsächlich - soll match vs. nicht tatsächlich - nicht soll vs. nicht tatsächlich

3.2 Wertverteilungen, Test auf Dopplungen

Lese Werte der Spalte Quellen als Liste aus.

```
In [32]: mainList = dfPoleis['Quellen'].values.tolist()
         mainList[0]
```

```
Out[32]: ['Diod. 14.90.3',
          'Diod. 19.65.6',
          'Diod. 14.78.5',
          'Diod. 14.90.3',
          'Diod. 19.65.6',
          'Diod. 19.110.4',
          'Manni  ( 1976',
          'Villard  ( 1954)',
          'Spigo  ( 1997',
          'Bertino  ( 1975)',
          'Bertino  ( 1975)',
          'Bertino  ( 1975)']
```

Reduziere Unterlisten auf eine Gesamtliste.

```
In [50]: quellenListe = []
         for sublist in mainList:
             if sublist:
                 for k in sublist:
                     quellenListe.append(k)
```

```
In [51]: quellenListe[:10]
```

```
Out[51]: ['Diod. 14.90.3',
          'Diod. 19.65.6',
          'Diod. 14.78.5',
          'Diod. 14.90.3',
          'Diod. 19.65.6',
          'Diod. 19.110.4',
          'Manni  ( 1976',
          'Villard  ( 1954)',
          'Spigo  ( 1997',
          'Bertino  ( 1975)']
```

Zähle die Häufigkeit der verschiedenen Quellen und speichere als Dictionary.

```
In [52]: quellenVerteilung = {x:quellenListe.count(x) for x in quellenListe}
         quellenVerteilung['Diod. 14.90.3']
```

```
Out[52]: 2
```

Erzeuge DataFrame, mit neuem Index und Namen der Spalten. Sortiere diesen Nach der Häufigkeit der Quelle.

```
In [88]: dfQuellenVerteilung = pd.DataFrame([quellenVerteilung])
         dfQuellenVerteilung
```

```
Out[88]:    Adamesteanu  ( 1986)  Adamesteanu  ( 1994  Agata  ( 1989)  Albini  ( 1964)  \
         0                     1                    1              2                1

            Allegro  ( 1991)  Allegro  ( 1997)  Angelis  ( 1994)  Anti  ( 1947)  \
         0                2                1                2             1

            Anti  ( 1981)  Antonaccio  ( 1997)        ...         Westermark  ( 1998)  \
         0             3                    4        ...                           2
```

```
   White  ( 1964)  Wilson  ( 1988)  Wilson  ( 1996)  Winter  ( 1963)  \
0              1               1               1               1

   Winter  ( 1971)  Yalouris  ( 1980)  Zahrnt  ( 1993)  Ziegler  ( 1934)  \
0              1              2              2              3

   Ziegler  ( 1943)
0              1

[1 rows x 567 columns]
```

In [89]:
```
dfQuellenVerteilung = dfQuellenVerteilung.transpose().reset_index()
dfQuellenVerteilung.head()
```

Out[89]:

	index	0
0	Adamesteanu (1986)	1
1	Adamesteanu (1994	1
2	Agata (1989)	2
3	Albini (1964)	1
4	Allegro (1991)	2

In [90]:
```
dfQuellenVerteilung = dfQuellenVerteilung.rename(columns={'index': 'Quelle', 0:'Hä
dfQuellenVerteilung.head()
```

Out[90]:

	Quelle	Häufigkeit
0	Adamesteanu (1986)	1
1	Adamesteanu (1994	1
2	Agata (1989)	2
3	Albini (1964)	1
4	Allegro (1991)	2

In [56]:
```
dfQuellenVerteilung.sort_values(by='Häufigkeit',ascending=False).head(10)
```

Out[56]:

	Quelle	Häufigkeit
382	Manganaro (1996	15
331	Hinz (1998)	13
509	Talbert (1974)	13
83	Cavalier (1991)	11
344	Karlsson (1995)	9
226	Diod. 14.78.7	9
113	Diod. 11.49.2	9
47	Boehringer (1998)	8
478	Rutter (1997)	8
327	Hansen (2000)	8

4 Größere Textblöcke, mit textblob and NLTK

In [34]:
```
from textblob import TextBlob
from textblob.taggers import NLTKTagger

from nltk.tokenize import SExprTokenizer
nltk_tagger = NLTKTagger()

import json
import pandas as pd
```

```
import re

import nltk.data
from nltk import PunktSentenceTokenizer
```

4.1 Alternative loading of data from online resources

Laden der gesamt Poleis Quell-Datei

```
In [35]: import requests # To communicate with websites
```

4.1.1 Github

```
In [36]: # The Github site for the lecture is public, therefore we can access the data in t
         PoleisDataOnline = requests.get('https://raw.githubusercontent.com/computational-h
         PoleisRawData2 = PoleisDataOnline.json()
         PoleisRawData2.keys()

Out[36]: dict_keys(['Ionia', 'Doris', 'Rhodos', 'Troas', 'Italia and Kampania', 'The Propor
```

4.1.2 edition-topoi.org

```
In [37]: PoleisDataOnline2 = requests.get('http://repository.edition-topoi.org/MISC/ReposMI
         PoleisRawData3 = PoleisDataOnline2.json()
         PoleisRawData3.keys()

Out[37]: dict_keys(['Ionia', 'Doris', 'Rhodos', 'Troas', 'Italia and Kampania', 'The Propor
```

4.2 Eigenschaften des Datasets PoleisRawData2

Each region contains the city names as second-level keys

```
In [38]: PoleisRawData2['Karia'].keys()

Out[38]: dict_keys(['Salmakis', 'Bolbai', 'Bargasa', 'Telemessos', 'Olymos', 'Telandros',
```

To access the text for a certain city, one has to use first and second level keys

```
In [39]: hydisosText = PoleisRawData2['Karia']['Hydisos']

In [40]: hydisosText

Out[40]: 'Identifier: 891. , (Hydisseus) Map  61.  Lat. 37.10,long.  27.50. Size  of  terri
```

4.2.1 Textblob

Generate blob using TextBlob. This allows identifying different parts of a text (words, sentences) or tagging words with their type (noun,verb,etc)

```
In [41]: blob = TextBlob(hydisosText)
```

Return noun phrases

```
In [45]: blob.noun_phrases

Out[45]: WordList(['identifier', 'hydisseus', 'lat', 'size', 'type', 'steph', 'byz', 'i.sti
```

Return tags of first 10 words, for definition see e.g. https://www.ling.upenn.edu/courses/Fall_2003/ling001/penn_treel

```
In [46]: blob.tags[:10]

Out[46]: [('Identifier', 'NN'),
          ('891.', 'CD'),
          ('Hydisseus', 'NNP'),
          ('Map', 'NNP'),
          ('61', 'CD'),
          ('Lat', 'NNP'),
          ('37.10', 'CD'),
          ('long', 'RB'),
          ('27.50', 'CD'),
          ('Size', 'NN')]
```

Return list of words

```
In [48]: blob.words

Out[48]: WordList(['Identifier', '891', 'Hydisseus', 'Map', '61', 'Lat', '37.10', 'long',
```

Return list of sentences

```
In [47]: blob.sentences

Out[47]: [Sentence("Identifier: 891. , (Hydisseus) Map  61."),
          Sentence("Lat."),
          Sentence("37.10,long."),
          Sentence("27.50."),
          Sentence("Size  of  territory:  ?"),
          Sentence("Type:  B:?"),
          Sentence("The  toponym  is  ` (Steph."),
          Sentence("Byz."),
          Sentence("645.17)."),
          Sentence("The  earliest  attestation  of  the  toponym is  in  a  C 1  inscriptio
          Sentence("4)."),
          Sentence("The city-ethnic  is  `  (IG i3  265.ii.51;Apollonius Aphrodisiensis  (F
          Sentence("4  (perhaps  C 3))  or  `  (I.Mylasa 401.8  (C 2C1))."),
          Sentence("Hydisos  was  a  member  of  the  Delian  League,  but  is  registered
          Sentence("At  the  site  of  Hydisos  there  are  remains  of  city  walls  and t
          Sentence("890."),
          Sentence("(Hymisseis) Map 61,  unlocated,  but  possibly  situat- ed  between  An
          Sentence("874)  and  Mylasa  (no."),
          Sentence("913)  (Pontani ( 1997)  7;  cf."),
          Sentence("L.  Robert  ( 1955)  226)."),
          Sentence("Type:  C: ."),
          Sentence("A  toponym  is not  attested."),
          Sentence("The  city-ethnic  is  `μ  (IG i3  262.iv.19; restored:  [h ]μ~) or `μ
          Sentence("127.58  (C 3))."),
          Sentence("The  collective  use  of  the  city-ethnic  is attested  externally  in
          Sentence("The  Hymisseis  were  members  of  the  Delian  League."),
          Sentence("They are  recorded  three  times  in  the  lists  from  451/0  (IG i3 2
          Sentence("In  the  assessment  decree  of 425/4  they  form  a  syntely  with the
          Sentence("892)  and  Kyromeis,  and  the  three  together are  assessed  at  6  t
          Sentence("(IG i3  71.ii.14344)."),
          Sentence("A  Hymesseus  is recorded  in a C 3  list  of  proxenoi  from  Eresos
          Sentence("796)  (IG xii suppl."),
          Sentence("127.58)."),
          Sentence("891. ")]
```

Not all sentences are correct. We have to use a different tokenizer to improve this.

By calling PunktSentenceTokenizer with an input text, we can train the detection of sentences.

This is usually a problem, since a lot of citations (parenthesis) or special characters hinder the detection of a sentence end.

To generate a text of all cities in a region we can use

```
In [52]: ioniaText = ''

         for key in PoleisRawData2['Ionia'].keys():
             ioniaText = ioniaText + (PoleisRawData2['Ionia'][key])
```

Train the tokenizer:

```
In [53]: trainedTokenizer = PunktSentenceTokenizer(ioniaText)
```

Apply it and print the results.

```
In [54]: for item in trainedTokenizer.tokenize(hydisosText):
             print(item)
             print("----")
```

```
Identifier: 891. , (Hydisseus) Map  61.  Lat. 37.10,long.
----
27.50. Size  of  territory:  ?
----
Type:  B:?
----
The  toponym  is  ` (Steph.
----
Byz. 645.17).
----
The  earliest  attestation  of  the  toponym is  in  a  C 1  inscription  (I.Stratonikeia
----
The city-ethnic  is  `  (IG i3  265.ii.51;Apollonius Aphrodisiensis  (FGrHist 740)  fr. 4
----
Hydisos  was  a  member  of  the  Delian  League,  but  is  registered  only  twice,  in
----
At  the  site  of  Hydisos  there  are  remains  of  city  walls  and towers,  probably  o
----
890.
----
(Hymisseis) Map 61,  unlocated,  but  possibly  situat- ed  between  Amyzon  (no.
----
874)  and  Mylasa  (no.
----
913)  (Pontani  ( 1997)  7;  cf.
----
L.  Robert  ( 1955)  226).
----
Type:  C: .
----
A  toponym  is not  attested.
----
The  city-ethnic  is  `μ  (IG i3  262.iv.19; restored:  [h ]μ~) or `μ  (IG i3  71.ii.143;
----
```

```
127.58  (C 3)).
----
The  collective  use  of  the  city-ethnic  is attested  externally  in  the  tribute  list
----
The  Hymisseis  were  members  of  the  Delian  League.
----
They are  recorded  three  times  in  the  lists  from  451/0  (IG i$^3$ 262.iv.19)  to  447/6
----
In  the  assessment  decree  of 425/4  they  form  a  syntely  with the  Edrieis  (no.
----
892)  and  Kyromeis,  and  the  three  together are  assessed  at  6  tal.  (IG i$^3$  71.ii.1
----
A  Hymesseus  is recorded  in a C 3  list  of  proxenoi  from  Eresos  (no.
----
796)  (IG xii suppl.
----
127.58).
----
891.
----
```

For citations, we need to find matching brackets. This can be done using SExprTokenizer.

```
In [58]: for i in SExprTokenizer(strict=False).tokenize(hydisosText):
             if i[0]=='(':
                 print(i)
```

```
(Hydisseus)
(Steph.  Byz. 645.17)
(I.Stratonikeia 508.10  (c. 81))
(FGrHist 740  fr. 4)
(IG i$^3$  265.ii.51;Apollonius Aphrodisiensis  (FGrHist 740)  fr. 4  (perhaps  C 3))
(I.Mylasa 401.8  (C 2C1))
(IG i$^3$  264.iii.21,  restored: `[~])
(IG i$^3$  265.ii.51,  restored: `[~])
(L.  Robert  ( 1935) 33940)
(Hymisseis)
(no. 874)
(no. 913)
(Pontani ( 1997)  7;  cf.  L.  Robert  ( 1955)  226)
(IG i$^3$  262.iv.19; restored:  [h ]$\mu$~)
(IG i$^3$  71.ii.143;  IG xii suppl. 127.58  (C 3))
(IG i$^3$  71.ii.143)
(IG i$^3$ 262.iv.19)
(IG i$^3$  265.ii.50)
(no. 892)
(IG i$^3$  71.ii.14344)
(no. 796)
(IG xii suppl. 127.58)
```

4.3 Create dataframe

```
In [119]: dfPoleisGesamt = pd.io.json.json_normalize(PoleisRawData2)
```

```
In [120]: dfPoleisGesamt= dfPoleisGesamt.transpose()
          dfPoleisGesamt.columns=['Beschreibung']
          dfPoleisGesamt.head(4)
```

Out[120]:

	Beschreibung
Achaia.Ascheion	Identifier: 233. , (Ascheieus) Unlocated. Typ...
Achaia.Boura	Identifier: 235. , (Bourios) Map 58. Lat. 38...
Achaia.Helike	Identifier: 236. , (Helikeus) Map 58. Lat. 3...
Achaia.Keryneia	Identifier: 237. , (Keryneus) Map 58. Lat. 3...

```
In [121]: dfPoleisGesamt= dfPoleisGesamt.reset_index()
          dfPoleisGesamt.head()
```

Out[121]:

	index	Beschreibung
0	Achaia.Ascheion	Identifier: 233. , (Ascheieus) Unlocated. Typ...
1	Achaia.Boura	Identifier: 235. , (Bourios) Map 58. Lat. 38...
2	Achaia.Helike	Identifier: 236. , (Helikeus) Map 58. Lat. 3...
3	Achaia.Keryneia	Identifier: 237. , (Keryneus) Map 58. Lat. 3...
4	Achaia.Leontion	Identifier: 238. , (Leontesios) Map 58.Lat.38...

```
In [122]: dfPoleisGesamt['indexSplit'] = dfPoleisGesamt['index'].str.split('.')
```

```
In [123]: dfPoleisGesamt['region'] = dfPoleisGesamt['indexSplit'].apply(lambda raw: raw[0])
          dfPoleisGesamt['city'] = dfPoleisGesamt['indexSplit'].apply(lambda raw: raw[1])
          dfPoleisGesamt.head()
```

Out[123]:

	index	Beschreibung	indexSplit	region	city
0	Achaia.Ascheion	Identifier: 233. , (Ascheieus) Unlocated. Typ...	[Achaia, Ascheion]	Achaia	Ascheion
1	Achaia.Boura	Identifier: 235. , (Bourios) Map 58. Lat. 38...	[Achaia, Boura]	Achaia	Boura
2	Achaia.Helike	Identifier: 236. , (Helikeus) Map 58. Lat. 3...	[Achaia, Helike]	Achaia	Helike
3	Achaia.Keryneia	Identifier: 237. , (Keryneus) Map 58. Lat. 3...	[Achaia, Keryneia]	Achaia	Keryneia
4	Achaia.Leontion	Identifier: 238. , (Leontesios) Map 58.Lat.38...	[Achaia, Leontion]	Achaia	Leontion

```
In [124]: dfPoleisGesamt = dfPoleisGesamt.drop('index', 1)
          dfPoleisGesamt = dfPoleisGesamt.drop('indexSplit', 1)
          dfPoleisGesamt.head()
```

Out[124]:

	Beschreibung	region	city
0	Identifier: 233. , (Ascheieus) Unlocated. Typ...	Achaia	Ascheion
1	Identifier: 235. , (Bourios) Map 58. Lat. 38...	Achaia	Boura
2	Identifier: 236. , (Helikeus) Map 58. Lat. 3...	Achaia	Helike
3	Identifier: 237. , (Keryneus) Map 58. Lat. 3...	Achaia	Keryneia
4	Identifier: 238. , (Leontesios) Map 58.Lat.38...	Achaia	Leontion

4.3.1 Get city identifier

Throughout the full text, cities are referenced by a running index. To make this information part of the dataframe, we extend it with an additional column.

```
In [125]: def cityIDFinder(text):
              '''
              #1: Find all occurance of the string "Identifier" followed by a colon, a spac
              #2: If there is a result, do the following
              #3: Take the first result idList[0] (because the identifier is at the beginni
                  and split the string at the dot (to remove the dot at the end of the stri
                  This ensures, that only a number is returned, since it removes the word
              '''
              idList = re.findall("Identifier\: \d{1,4}\.", text) #1
              if idList: #2
                  idCity = idList[0].split('.')[0][12:] #3
                  return idCity

In [126]: dfPoleisGesamt['city_id'] = dfPoleisGesamt['Beschreibung'].apply(lambda row: city
```

4.3.2 Collection of all citations

To collect all citations in the text for one city, we first use a tokenizer from NLTK. This tokenizer collects all parenthesis and is much easier to use, that regular expressions.

The basic assumption for citations is: They are written in parenthesis, start with a capital letter, and contain at least one blank space (to separate the authors name from text pages, indices, or dates).

```
In [127]: def citationFinder(text):
              '''
              #1: Generate a list of all capital letters
              #2: Tokenize text to search for parenthesis, '( ... )' (this returns more acc
                  The option strict=False is necessary to prevent errors, when the text con
              #3: The basic assumptions for citations are:
                  In parenthesis (first element is opening paranthesis),
                  start with a capital letter (second element is a capital letter),
                  and contain at least one blank space ' '
              '''
              import string
              letters=[i for i in string.ascii_uppercase] #1
              paranthesisTokenized = SExprTokenizer(strict=False).tokenize(text) #2
              listCite = [x for x in paranthesisTokenized if x[0] == '(' and x[1] in letter
              return listCite

In [128]: dfPoleisGesamt['sources'] = dfPoleisGesamt['Beschreibung'].apply(lambda row: cita
```

4.3.3 Transformation of coordinates

A simple regular expression is enough to find all coordinates in the text. The coordinates are transformed from degrees/minutes to decimal to enable plotting on a map with common projection.

```
In [129]: def coordinateFinder(value,pattern):
              '''
              #1: General function for finding regular expression pattern in a text.
              #2: If patterns are found, do the following
              #3: Take the last five values of the first string returned
              #4: To convert angular in decimal coordinates:
                  Take the returned value, split it at the dot
                  convert the first part into a floating number (e.g. 36.0),
                  and the second part into a integer number (e.g. 34) and divide it by 60.
                  The sum the two results to return a coordinate in decimal system
```

```
            '''
            x = re.findall(pattern, value)                                    #
            if x:                                                             #
                coord = x[0][-5:]                                             #
                decCord = float(coord.split('.')[0]) +  int(coord.split('.')[-1])/60   #
                return decCord
```

```
In [130]: dfPoleisGesamt['latitude'] = dfPoleisGesamt["Beschreibung"].apply(coordinateFinde
          dfPoleisGesamt['longitude'] = dfPoleisGesamt["Beschreibung"].apply(coordinateFin
```

4.3.4 Proper nouns

To generate a list of all mentioned proper nouns for each city, we use TextBlob. TextBlob is a NLTK tool with parts-of-speech tagger. We are interessted in all parts that are 'NNP' and longer then 3 letters.

This takes some time to process for the full dataframe. Behaviour can be tested by uncommenting the cell below.

```
In [131]: def namesFinder(text):
              '''
              #1: Generate a blob out of the text
              #2: Generate a list of all POS Tags, that are labeld as NNP(S) (Proper noun,
              '''
              blobs = TextBlob(text)
              namesList = [x[0] for x in blobs.pos_tags if (x[1] == 'NNP') | (x[1] == 'NNPS
              return namesList
```

```
In [134]: # Uncomment to test routine.

          namesFinder(dfPoleisGesamt['Beschreibung'].iloc[10])
```

```
Out[134]: ['Herodotos',
           'Stein',
           'Aigiroessa',
           'Elaia',
           'Herodotos',
           'Elaia',
           'Head',
           'Cook',
           'Aigiroessa',
           'Belkahve']
```

```
In [135]: ########################################
          # Careful: Takes some time to evalute! #
          ########################################

          dfPoleisGesamt['names'] = dfPoleisGesamt['Beschreibung'].apply(lambda row: namesF
```

4.3.5 Cross links to other cities

Links to other cities are mentioned in the fulltext with reference to the index (e.g. '(no. 982)'). searching for these should give a link list.

```
In [136]: def linksFinder(text):
              '''
              #1: Find all occurances of the string "(no. 1234)" with between one and four
              #2: If we have a result
```

```
        #3: Generate a list, where every result:
            is split at the space, take the last part, and only up to the last lette
        #4: For all these results, convert the entries into an integer number
        '''
        x = re.findall("\(no\. \d{1,4}\)", text)                #1
        if x:                                                   #2
            links = [((z.split(' '))[-1])[:-1] for z in x]      #3
            linksInt = [int(x) for x in links]                  #4
            return linksInt
```

```
In [137]: dfPoleisGesamt['linkedCities'] = dfPoleisGesamt['Beschreibung'].apply(lambda row
```

4.4 Display dataframe

```
In [138]: # Uncomment to display full dataframe
          #df
          dfPoleisGesamt.head(4)
```

Out[138]:

	Beschreibung	region	city	\
0	Identifier: 233. , (Ascheieus) Unlocated. Typ...	Achaia	Ascheion	
1	Identifier: 235. , (Bourios) Map 58. Lat. 38...	Achaia	Boura	
2	Identifier: 236. , (Helikeus) Map 58. Lat. 3...	Achaia	Helike	
3	Identifier: 237. , (Keryneus) Map 58. Lat. 3...	Achaia	Keryneia	

	city_id	sources	latitude	\
0	233	[(CID ii 51.8 (339/8)), (BCH 45 (1921) i...	NaN	
1	235	[(Morgan and Hall (1996) 175; Barr.), (R...	38.166667	
2	236	[(Morgan and Hall (1996) 175; Barr.), (Dio...	38.250000	
3	237	[(Rizakis (1995) 206; Barr.), (Paus. 7.25...	38.166667	

	longitude	names	\
0	NaN	[Ascheieus, F.Delphes, Delphic, F.Delphes, Asc...	
1	22.250000	[Bourios, Keryneia, Paus, Strabo, Boura, Diako...	
2	22.166667	[Helikeus, Paus, Aigion, Strabo, Herakleides, ...	
3	22.166667	[Keryneus, Paus, Keryneia, Mamousia, Derveni, ...	

	linkedCities
0	None
1	[236, 235, 238, 251, 165, 148]
2	[231, 70]
3	[353]