Entscheidungsdatenbank auf regulären Ausdruck durchsuchen

Beispiel hier: 'spekulativ' und verwandte Wörter - für andere Suchbegriffe `regex`, `searchterm` und ggf. die Displayvariablen anpassen.

Getestet mit `sqlite 3.13.0`, `networkx 1.11`, `matplotlib 2.0.2`, `numpy 1.12.0` und `seaborn 0.8.1`.

In [ ]:

```
import sqlite3, re

def search_decision_texts(path_to_db, regex):
    con = sqlite3.connect(path_to_db)
    cur = con.cursor()
    sql = '''SELECT aktenzeichen, entschdatum, volltext
             FROM entscheidungen
             ORDER BY date(entschdatum)'''
    matches = {}
    for row in con.execute(sql):
        match = re.findall(regex, row[2], re.DOTALL)
        if match:
            matches[(row[1], row[0])] = len(match)
    con.close()
    return matches

path_to_db = '../data/bgh.db'
regex = r'.{0,20}[Ss]-?\s*p-?\s*-?e-?\s*k-?\s*u-?\s*l.{0,20}'

matches = search_decision_texts(path_to_db, regex)

#       = {('1988-10-11', 'XI ZR 67/88'): 2,
#          ('1989-02-28', 'XI ZR 70/88'): 1,
#          ('1989-04-18', 'XI ZR 133/88'): 1,
#            ...
#         }
```

In [ ]:

```
import networkx as nx, matplotlib.pyplot as plt, numpy as np, seaborn as sns
```

In [ ]:

```
searchterm = 'spekulativ'
startyear = 1988
endyear = 2015
yeardist = 30
monthdist = 31
```

In [ ]:

```
G = nx.Graph()
for match in matches.keys():
    G.add_node(match)
    G.node[match]['az'] = re.search(r'\d{1,3}/\d{2}', match[1]).group(0)
    G.node[match]['verfahren'] = re.search('(?<=XI )..', match[1]).group(0)
    G.node[match]['jahr'] = int(match[0][:4])
    G.node[match]['monat'] = int(match[0][5:7])
    G.node[match]['tag'] = int(match[0][8:10])
    G.node[match]['x'] = (G.node[match]['jahr']-startyear)*yeardist
    G.node[match]['y'] = ((G.node[match]['monat']-1)*monthdist
                          +G.node[match]['tag'])*2
    G.node[match][f'{searchterm}'] = matches[match]
```

In [ ]:

```
%matplotlib notebook
```

In [ ]:

```
sns.set_style('whitegrid')
```

In [ ]:

```
plt.rcParams['figure.figsize'] = (9,6)
plt.rcParams['font.serif'] = 'BitstreamVeraSans Roman'
plt.rcParams['font.family'] = 'serif'
plt.rcParams['font.style'] = 'normal'
plt.rcParams['font.size'] = 6.5
```

In [ ]:

```
xticks = [n*yeardist for n in range(0,(endyear-startyear)+2)]
xticklabels = [str(n) for n in range(startyear,endyear+1)]
yticks = [n*31*2 for n in range(0,13)]
yticklabels = ['Jan','Feb','Mär','Apr','Mai','Jun',
               'Jul','Aug','Sep','Okt','Nov','Dec']
```

In [ ]:

```
xs = [G.node[n]['x'] for n in G.nodes()]
ys = [G.node[n]['y'] for n in G.nodes()]
sizes = [G.node[n][f'{searchterm}'] for n in G.nodes()]
minsize, maxsize = 5, 20
maxmatches = max(matches.values())
fontsizes = np.linspace(minsize,maxsize,maxmatches)
txt = ['$'+G.node[n]['az']+'$' for n in G.nodes()]
colors = ['k' if G.node[n]['verfahren'] == 'ZR' else (
    'red' if G.node[n]['verfahren'] == 'ZB' else 'blue') for n in G.nodes()]

plt.scatter(xs,ys,s=sizes, marker='')
plt.xlim(xticks[0],xticks[-1])
plt.ylim(yticks[0],yticks[-1])
plt.xticks(xticks,xticklabels)
plt.yticks(yticks,yticklabels)
plt.gca().get_xaxis().set_tick_params(which='major', pad=7.5)
plt.gca().get_yaxis().set_tick_params(which='major', pad=7.5)
for i, txt in enumerate(txt):
    plt.text(xs[i], ys[i], txt,
             fontdict=dict(fontsize=fontsizes[sizes[i]-1], color=colors[i],
             verticalalignment='baseline', horizontalalignment='left'))
```

Falls die Graphik extern gespeichert werden soll:

In [ ]:

```
plt.savefig(f'../graphics/{searchterm}.png', dpi=1200) # Warnung kann ignoriert
 werden
```

In [ ]:

```
plt.savefig(f'../graphics/{searchterm}.svg', format='svg') # Warnung kann ignori
ert werden
```

Ende.