IM-UH 1511 **Introduction to Digital Humanities**

HOMEWORK 2

Network of Co-Occurring Words (Word-Net) in Sentences of the Text

50 points totally

In [1]:
```
import time
start_time = time.perf_counter()
import urllib, os, codecs, random, operator, re, string, copy, dateutil.par
from collections import Counter
from string import punctuation, digits
import pathlib
import spacy
from spacy import displacy
nlp = spacy.load('en_core_web_lg')
import inflect
import nltk
from nltk import word_tokenize
import pygraphviz
from networkx.drawing.nx_agraph import graphviz_layout
import networkx as nx
from wordcloud import WordCloud
import matplotlib as mpl
from nltk.tokenize import word_tokenize
from nltk.tokenize import sent_tokenize
from textblob import TextBlob

import warnings
warnings.filterwarnings("ignore", category=RuntimeWarning)
warnings.simplefilter('ignore')
```

Load Data

In [2]:
```
# get your working directory
home = str(pathlib.Path.cwd())

# create a path to which the file will be written
text_path = os.path.join(home, 'WizardofOz.txt')

# location of the project gutenberg copy of the moby-dick text file
text_url = 'http://www.gutenberg.org/cache/epub/55/pg55.txt'

urllib.request.urlretrieve(text_url, text_path)

print('Downloaded to:', text_path)
```

```
Downloaded to: /Users/mb7881/WorkPlaces/Python Projects 2/3 NYUAD Digital
Humanities/Homework2 NetworkOfNames/WizardofOz.txt
```

In [3]:
```
f = codecs.open(text_path, "r", encoding="utf-8").readlines()
for line in f:
    if line.startswith("1.  The Cyclone"):
        print(f.index(line)) #198
    if line.startswith("And oh, Aunt Em!  I'm so glad to be at home again!"
        print(f.index(line)) #15514
```

```
109
4753
```

In [4]:
```
ff=f[109:4756]
ff
```

Out[4]:
```
['1.  The Cyclone\r\n',
 '\r\n',
 '\r\n',
 'Dorothy lived in the midst of the great Kansas prairies, with Uncle\r
\n',
 "Henry, who was a farmer, and Aunt Em, who was the farmer's wife.  Their
\r\n",
 'house was small, for the lumber to build it had to be carried by wagon
\r\n',
 'many miles.  There were four walls, a floor and a roof, which made one
\r\n',
 'room; and this room contained a rusty looking cookstove, a cupboard for
\r\n',
 'the dishes, a table, three or four chairs, and the beds.  Uncle Henry\r
\n',
 'and Aunt Em had a big bed in one corner, and Dorothy a little bed in\r
\n',
 'another corner.  There was no garret at all, and no cellar--except a\r
\n',
```

In [5]:
```
ff[-10:]
```

Out[5]:
```
['Aunt Em had just come out of the house to water the cabbages when she\r
\n',
 'looked up and saw Dorothy running toward her.\r\n',
 '\r\n',
 '"My darling child!" she cried, folding the little girl in her arms and
\r\n',
 'covering her face with kisses.  "Where in the world did you come fro
m?"\r\n',
 '\r\n',
 '"From the Land of Oz," said Dorothy gravely.  "And here is Toto, too.\r
\n',
 'And oh, Aunt Em!  I\'m so glad to be at home again!"\r\n',
 '\r\n',
 '\r\n']
```

In [6]:
```
text="\r\n".join(ff)
text
```

Out[6]:
```
'1.  The Cyclone\r\n\r\n\r\n\r\n\r\n\r\n\r\nDorothy lived in the midst of the
great Kansas prairies, with Uncle\r\n\r\nHenry, who was a farmer, and Aun
t Em, who was the farmer\'s wife.  Their\r\n\r\nhouse was small, for the
lumber to build it had to be carried by wagon\r\n\r\nmany miles.  There w
ere four walls, a floor and a roof, which made one\r\n\r\nroom; and this
room contained a rusty looking cookstove, a cupboard for\r\n\r\nthe dishe
s, a table, three or four chairs, and the beds.  Uncle Henry\r\n\r\nand A
unt Em had a big bed in one corner, and Dorothy a little bed in\r\n\r\nan
other corner.  There was no garret at all, and no cellar--except a\r\n\r
\nsmall hole dug in the ground, called a cyclone cellar, where the family
\r\n\r\ncould go in case one of those great whirlwinds arose, mighty enou
gh to\r\n\r\ncrush any building in its path.  It was reached by a trap do
or in the\r\n\r\nmiddle of the floor, from which a ladder led down into t
he small, dark\r\n\r\nhole.\r\n\r\n\r\n\r\nWhen Dorothy stood in the door
way and looked around, she could see\r\n\r\nnothing but the great gray pr
airie on every side.  Not a tree nor a\r\n\r\nhouse broke the broad sweep
of flat country that reached to the edge of\r\n\r\nthe sky in all directi
ons.  The sun had baked the plowed land into a\r\n\r\ngray mass, with lit
tle cracks running through it.  Even the grass was\r\n\r\nnot green, for
```

In [7]:
```
titlename = "L. Frank Baum's The Wonderful Wizard of Oz"

words = word_tokenize(text)
nuw=len(words)
uw=len(set(words))
print("%s contains %i nonunique and %i unique words"%(titlename,nuw,uw))
```

```
L. Frank Baum's The Wonderful Wizard of Oz contains 46908 nonunique and 3
182 unique words
```

Extraction of Proper Nouns

In [8]:
```
p = inflect.engine()
d_tags = {}

docs_d={"WizardofOz":text}
for key, value in docs_d.items():
    arr = []
    doc = nlp(value.replace('\n',''))
    #Keep these types of nlp entities
    keep_l = ['PERSON'] #,'NORP','PRODUCT','ORG']
    #Typo/model error + german corrections
    drop_t = []

    #Things inflect library handles poorly or to exclude from touching
    ex_ls = []

    for X in doc.ents:
        s1 = X.text
        if (X.label_ in keep_l) and (s1.lower() not in drop_t) and (s1):
            arr.append((s1, X.label_))
    d_tags[key] = arr
# pprint(d_tags)
names=[]
for k,v in d_tags.items():
    for vv in v:
        if vv[0] not in names:
            p=vv[0].replace("'","")
            p=p.title()
            names.append(p)
names=sorted(set(names))
print(len(names))
names
```

```
66
```

In [9]:
```
rem=[]
for p in names:
    if "_" in p:
        rem.append(p)
    if "--" in p:
        rem.append(p)
    if p not in text:
        rem.append(p)
names=[p for p in names if p not in rem]
pp=[q for q in itertools.product(names,names) if q[0]!=q[1]]
for q in pp:
    if q[0] in q[1]:
        rem.append(q[0])
    if q[1] in q[0]:
        rem.append(q[1])
    w=q[0]+" "+q[1]
    if w in text:
        names.append(w)
        rem.append(q[0])
        rem.append(q[1])
names=[p for p in names if p not in rem]
names=sorted(set(names))
print(len(names))
sorted(names)
```

```
27
```

Out[9]:
```
['Aunt Em',
 'Boq',
 'Gates',
 'Gayelette',
 'Hammer',
 'Head Dorothy',
 'Joker',
 'Kalidahs',
 'Lady',
 'Meek',
 'Munchkin',
 'Quadlings',
 'Queen',
 'Quelala',
 'Scarecrow',
 'Sorrowfully Dorothy',
 'Stork',
 'The Cowardly Lion'
```

In [10]:
```
rem=['Wizard Oz', 'Gates', 'Hammer', 'Head Dorothy', 'Kalidahs', 'Lady', 'M
    'tin man', 'Joker', 'This Golden Cap', 'The Wicked Witch','The Cowardly
    'The King Crow', 'Stork', 'Good Witch of the North', 'Quadlings',"The M
     "The Witch","Guardian"]
names=[p for p in names if p not in rem]
names=names+['Tin Woodman', 'Scarecrow', 'Oz', 'Guardian of the Gates', 'Mr
             'Wicked Witch of the West', 'Glinda', 'Dorothy', 'Lion', 'King
             'Witch of the North', 'Hammer-Heads', 'Quadlings', 'Winkies',
            'green girl', 'Monkey King', 'Munchkins', 'Cowardly Lion', 'Wil
             'Wizard',"Great Oz","Mouse"]


names=sorted(set(names))
print(len(names))
names
```

```
35
```

Out[10]:
```
['Aunt Em',
 'Boq',
 'Cowardly Lion',
 'Dorothy',
 'Gayelette',
 'Glinda',
 'Great Oz',
 'Great Wizard',
 'Guardian of the Gates',
 'Hammer-Heads',
 'Kalidahs',
 'King Crow',
 'King of the Winged Monkeys',
 'Lion',
 'Monkey King',
 'Mouse',
 'Mr. Joker',
 'Munchkins',
 'Oz',
 'Quadlings',
 'Queen',
 'Quelala',
 'Scarecrow',
 'Stork',
 'Tin Woodman',
 'Toto',
 'Uncle Henry',
 'Wicked Witch of the East',
 'Wicked Witch of the West',
 'Wildcat',
 'Winkies',
 'Witch of the North',
 'Wizard',
 'green girl',
 'tin man']
```

In [11]:
```
nfreq=[]
for i in names:
    nfreq.append(text.count(i))
pnf_df = pd.DataFrame(
    {'Proper Nouns': names,
     'Frequency of Occurrences': nfreq
    })
pnf_df=pnf_df[['Proper Nouns','Frequency of Occurrences']]
pnf_df=pnf_df.sort_values(by ='Frequency of Occurrences',ascending=False)
# trf_df=trf_df[trf_df["Frequency of Occurrences"]>10]
print(len(pnf_df))
pnf_df[:50]
```

```
35
```

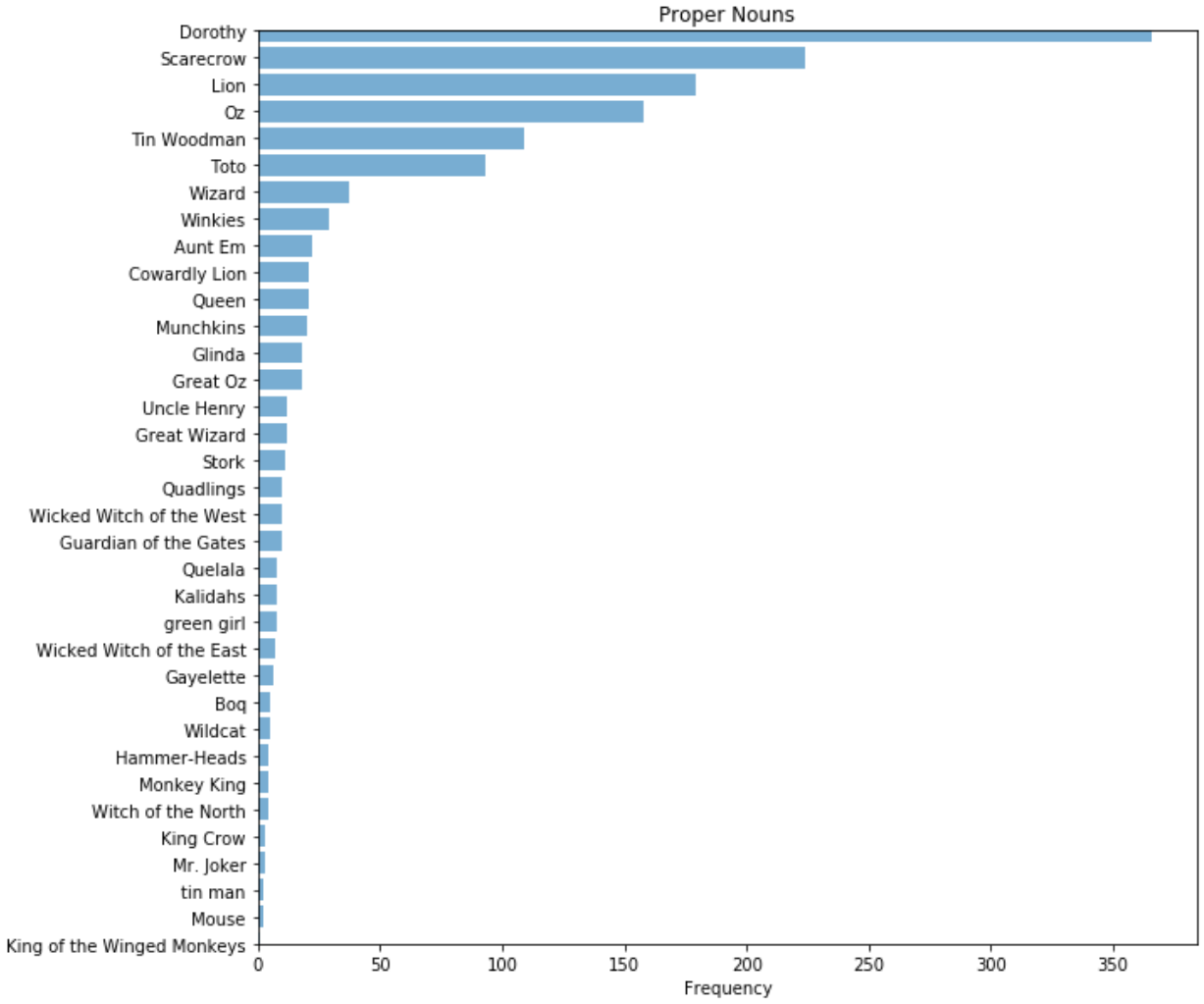
Out[11]:

	Proper Nouns	Frequency of Occurrences
3	Dorothy	366
22	Scarecrow	224
13	Lion	179
18	Oz	158
24	Tin Woodman	109
25	Toto	93
32	Wizard	37
30	Winkies	29
0	Aunt Em	22
2	Cowardly Lion	21
20	Queen	21
17	Munchkins	20
5	Glinda	18
6	Great Oz	18
26	Uncle Henry	12
7	Great Wizard	12
23	Stork	11
19	Quadlings	10
28	Wicked Witch of the West	10
8	Guardian of the Gates	10
21	Quelala	8
10	Kalidahs	8
33	green girl	8
27	Wicked Witch of the East	7
4	Gayelette	6
1	Boq	5

	Proper Nouns	Frequency of Occurrences
29	Wildcat	5
9	Hammer-Heads	4
14	Monkey King	4
31	Witch of the North	4
11	King Crow	3
16	Mr. Joker	3
34	tin man	2
15	Mouse	2
12	King of the Winged Monkeys	1

In [12]:
```
pnf_df.to_csv('Names_freqs.csv')
```

In [13]:
```
x = pnf_df.set_index('Proper Nouns').T.to_dict()
x=sorted([(k,v['Frequency of Occurrences']) for k,v in x.items()], key=lamb
keys = [i for (i,j) in x]
y_pos = np.arange(len(keys))
performance = [j for (i,j) in x]
plt.figure(figsize=(10,10))
ax = plt.axes()
plt.barh(y_pos, performance, align='center', alpha=0.6)
ax.invert_yaxis()
plt.yticks(y_pos, keys)
plt.xlabel('Frequency')
plt.title('Proper Nouns')
plt.show()
```



In [14]:
```
t=[]
for (i,j) in x:
    for k in range(j):
#         print i.replace(" ","_").replace("-","_")
        t.append(i.replace(" ","_").replace("-","_"))
ttd=' '.join(t)
wordcloud = WordCloud(collocations=False,background_color="white",colormap=
fig = plt.figure(figsize=(13,13))
default_colors = wordcloud.to_array()
plt.imshow(default_colors, interpolation="bilinear")
plt.axis("off")
ss="Proper Nouns of %s" %titlename
plt.suptitle(ss,fontsize=25)
plt.tight_layout(rect=[0, 0, 1, 1.4])
plt.show()
```

Dictionary of Aliased Proper Nouns

In [15]:
```
alias_dict={}
for n in names:
    if n=="Tin Woodman":
        alias_dict[n]="tin man"
    elif n=="tin man":
        alias_dict[n]="tin man"
    elif n=="Oz":
        alias_dict[n]="Oz"
    elif n=="Great Oz":
        alias_dict[n]="Oz"
    elif n=="Wizard":
        alias_dict[n]="Oz"
    elif n=="Great Wizard":
        alias_dict[n]="Oz"
    elif n=="Cowardly Lion":
        alias_dict[n]="Lion"
    elif n=="Lion":
        alias_dict[n]="Lion"
    elif n=="King of the Winged Monkeys":
        alias_dict[n]="Monkey King"
    elif n=="Monkey King":
        alias_dict[n]="Monkey King"
    else:
        alias_dict[n]=n
print("The dictionary of aliases has %i keys (names) and %i unique values (
# for k,v in alias_dict.items():
#     print(k,"-->",v)
```

```
The dictionary of aliases has 35 keys (names) and 29 unique values (alias
ed proper nouns)
```

The Network of Sententially Co-Occurring Proper Names ("Word-Net")

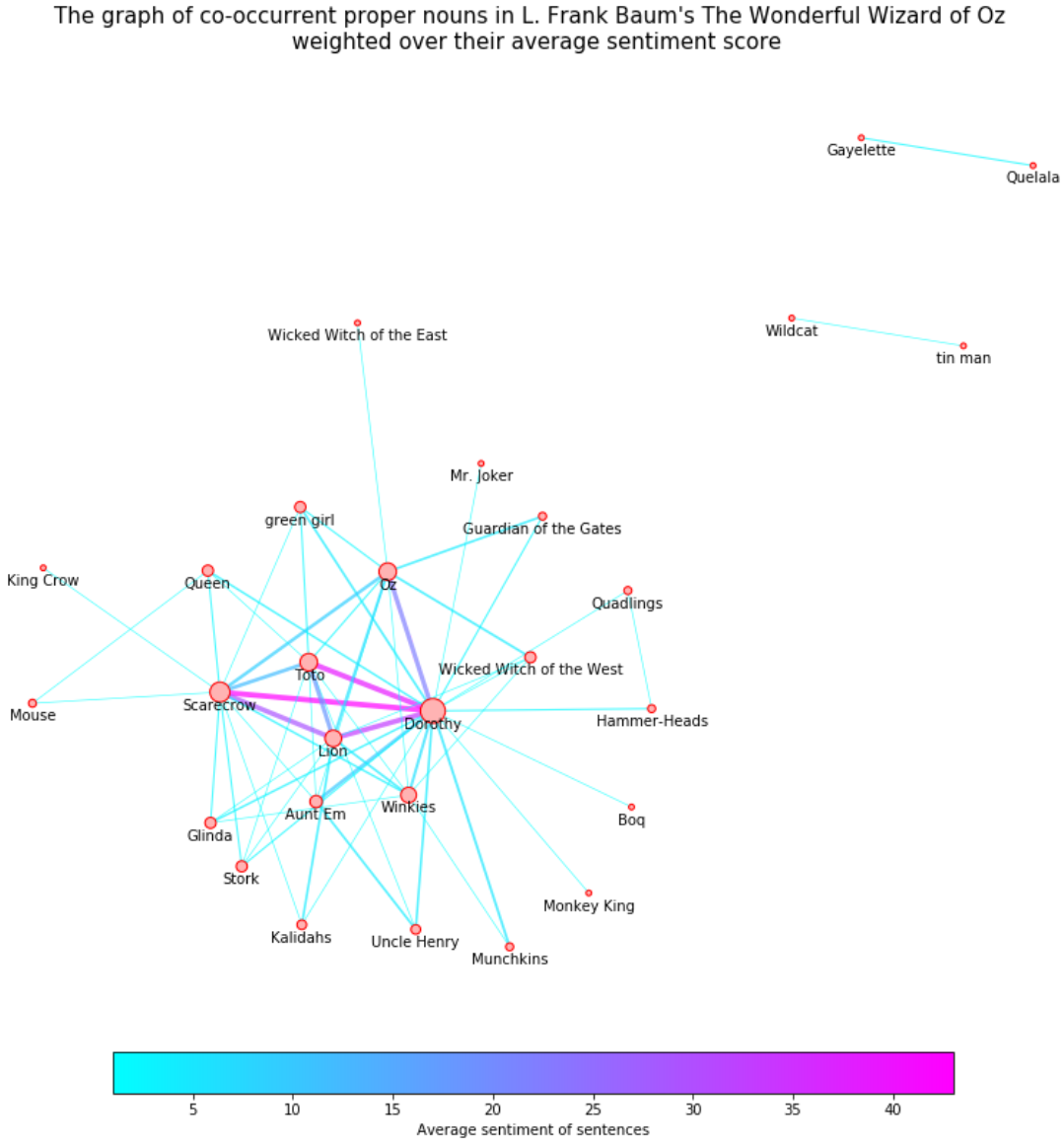
In [16]:
```
blob = TextBlob(text)
textSentences = blob.sentences
sendic=dict()
for i,v in enumerate(textSentences):
    sent=v.sentiment.polarity
    wl=[]
    for term in list(set(alias_dict.values())):
        if term in v:
            wl.append(term)
    if len(wl)>1:
        sendic[i]=wl
medges=[]
for k,v in sendic.items():
    sent=textSentences[k].sentiment.polarity
    dd={}
    ps=set()
    for j in itertools.combinations(v, 2):
        ps.add(j)
        dd[j]=(k,sent)
    for jj in ps:
        s=0
        ss=0
        for kk,vv in dd.items():
            if kk==jj:
                s+=1
                ss+=vv[1]
        if alias_dict[jj[0]]!=alias_dict[jj[1]]:
            medges.append((alias_dict[jj[0]],alias_dict[jj[1]],"Sentence_"+
print("%s contains %i sentential co-occurrences among %i aliased proper nou
medges
```

```
L. Frank Baum's The Wonderful Wizard of Oz contains 317 sentential co-occ
urrences among 29 aliased proper nouns
```

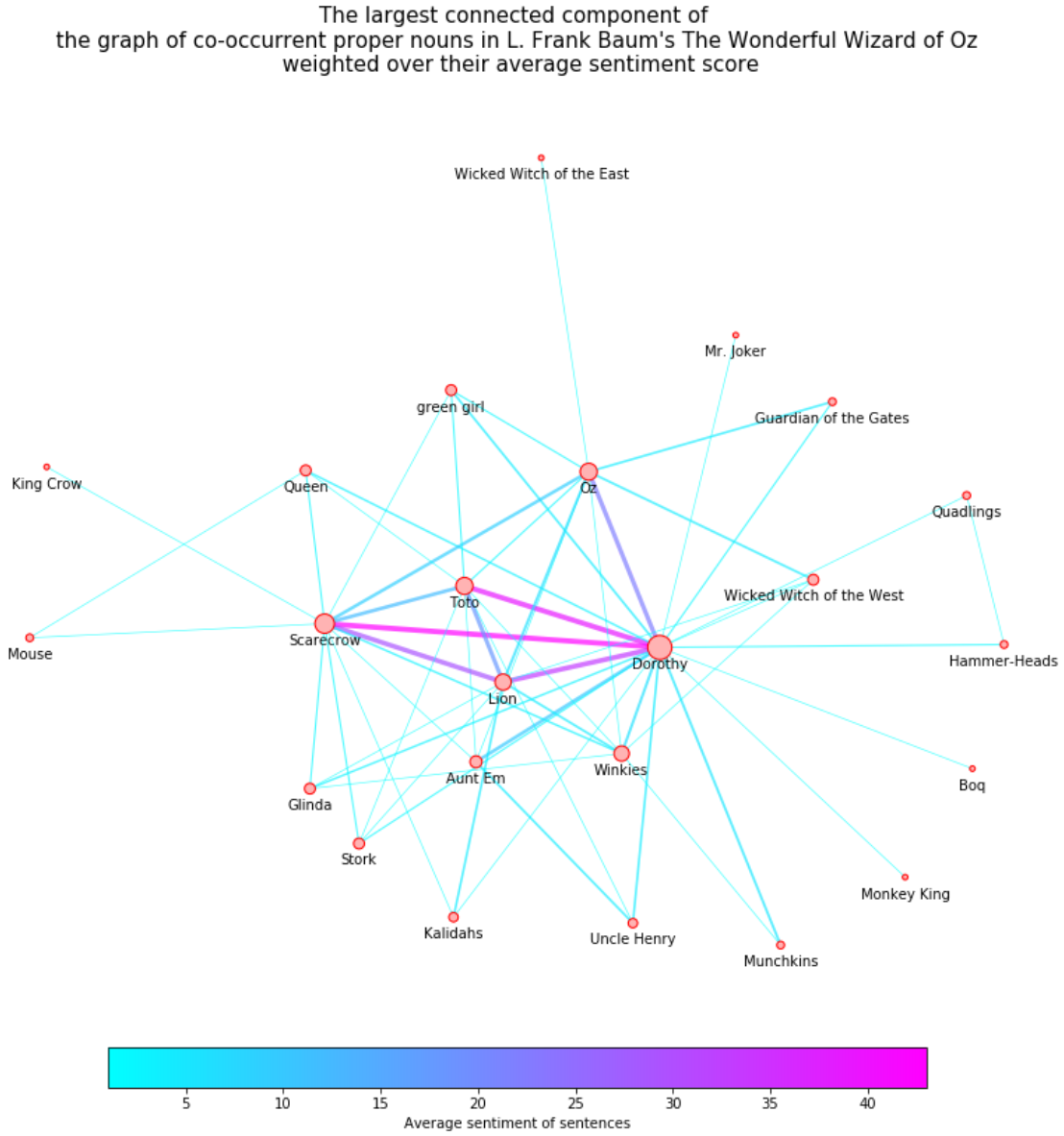
In [17]:
```
medgesd=[]
for e in medges:
    d={}
    d['Sentence']=e[2]
    d['Average sentiment']=e[3]
    medgesd.append((e[0],e[1],d))

G = nx.MultiGraph()
G.add_edges_from(medgesd)
for e in G.edges(data=True):
    if e[0]==e[1]:
        G.remove_edge(e[0],e[1])
weight={(x,y):v for (x, y), v in Counter(G.edges()).items()}
w_edges=[(x,y,z) for (x,y),z in weight.items()]
Gw = nx.Graph()
Gw.add_weighted_edges_from(w_edges)

print("The graph of sententially co-occurrent proper nouns in %s is a weigh
out=' '.join([n+"\n" for n in alias_dict.values() if n not in Gw.nodes()])
print("The proper names which do not co-occur in sentences are: \n %s" %out
# print "Graph Gw is a weighted graph with %i nodes and %i edges" %(len(Gw.
print("The density of this graph is %.3f" %nx.density(Gw))
if nx.is_connected(Gw)==True:
    print ("This graph is a connected graph")
else:
    print ("This graph is a disconnected graph and it has",nx.number_connec
giant = max(nx.connected_component_subgraphs(Gw), key=len)
Gwlcc=Gw.subgraph(giant)
print ("The largest connected component of this graph is a weighted graph w
print ("The density of the largest connected component of this graph is %.3
```

```
The graph of sententially co-occurrent proper nouns in L. Frank Baum's Th
e Wonderful Wizard of Oz is a weighted graph and it has 28 nodes and 60 e
dges

The proper names which do not co-occur in sentences are:
 Witch of the North

The density of this graph is 0.159
This graph is a disconnected graph and it has 3 connected components
The largest connected component of this graph is a weighted graph with 24
nodes and 58 edges
The density of the largest connected component of this graph is 0.210
```

In [18]:
```
edge_width=[Gw[u][v]['weight'] for u,v in Gw.edges()]
edge_width=[math.log(1+w) for w in edge_width]
cmap=plt.cm.cool
weight_list = [ e[2]['weight'] for e in Gw.edges(data=True) ]
edge_color=weight_list
vmin = min(edge_color)
vmax = max(edge_color)
# width_list=[2*math.log(2+w) for w in weight_list]
width_list=[1.5*math.log(abs(min(weight_list))+2+w) for w in weight_list] #
nsi=[15*Gw.degree(n) for n in Gw.nodes()]

figsize=(15,15)

pos=graphviz_layout(Gw)
# pos=nx.spring_layout(Gw)

node_color="#ffb3b3"
node_border_color="r"
plt.figure(figsize=figsize);
nodes = nx.draw_networkx_nodes(Gw, pos, node_color=node_color,node_size=nsi
nodes.set_edgecolor(node_border_color)
nx.draw_networkx_edges(Gw, pos, edge_color=edge_color,edge_cmap=cmap,vmin=v
plt.axis('off');
yoffset = {}
y_off = -5 # offset on the y axis
for k, v in pos.items():
    yoffset[k] = (v[0], v[1]+y_off)
nx.draw_networkx_labels(Gw, yoffset,font_size=10);
sm = plt.cm.ScalarMappable(cmap=cmap, norm=plt.Normalize(vmin=vmin, vmax=vm
sm.set_array([])
cbar = plt.colorbar(sm, orientation='horizontal', shrink=0.7, pad = 0.02)
cbar.set_label('Average sentiment of sentences')
sst="The graph of co-occurrent proper nouns in %s \n weighted over their av
plt.title(sst,fontsize=15);
plt.margins(x=0.1, y=0.1)
```


The graph of co-occurrent proper nouns in L. Frank Baum's The Wonderful Wizard of Oz
weighted over their average sentiment score
Gayelette
Quelala
Wicked Witch of the East
Wildcat
tin man
Mr. Joker
green girl
Guardian of the Gates
King Crow
Queen
Oz
Quadlings
Toto
Wicked Witch of the West
Mouse
Scarecrow
Dorothy
Hammer-Heads
Lion
Winkies
Aunt Em
Boq
Glinda
Stork
Monkey King
Kalidahs
Uncle Henry
Munchkins
5
10
15
20
25
30
35
40
Average sentiment of sentences

In [19]:
```
Gw=Gwlcc

edge_width=[Gw[u][v]['weight'] for u,v in Gw.edges()]
edge_width=[math.log(1+w) for w in edge_width]
cmap=plt.cm.cool
weight_list = [ e[2]['weight'] for e in Gw.edges(data=True) ]
edge_color=weight_list
vmin = min(edge_color)
vmax = max(edge_color)
# width_list=[2*math.log(2+w) for w in weight_list]
width_list=[1.5*math.log(abs(min(weight_list))+2+w) for w in weight_list] #
nsi=[15*Gw.degree(n) for n in Gw.nodes()]

figsize=(15,15)
pos=graphviz_layout(Gw)
node_color="#ffb3b3"
node_border_color="r"
plt.figure(figsize=figsize);
nodes = nx.draw_networkx_nodes(Gw, pos, node_color=node_color,node_size=nsi
nodes.set_edgecolor(node_border_color)
nx.draw_networkx_edges(Gw, pos, edge_color=edge_color,edge_cmap=cmap,vmin=v
plt.axis('off');
yoffset = {}
y_off = -5 # offset on the y axis
for k, v in pos.items():
    yoffset[k] = (v[0], v[1]+y_off)
nx.draw_networkx_labels(Gw, yoffset,font_size=10);
sm = plt.cm.ScalarMappable(cmap=cmap, norm=plt.Normalize(vmin=vmin, vmax=vm
sm.set_array([])
cbar = plt.colorbar(sm, orientation='horizontal', shrink=0.7, pad = 0.02)
cbar.set_label('Average sentiment of sentences')
sst="The largest connected component of \n the graph of co-occurrent proper
plt.title(sst,fontsize=15);
plt.margins(x=0.1, y=0.1)
```



In [20]:
```
print("Run in %.2f seconds (%.2f minutes)" %(time.clock() - start_time,(tim
```

```
Run in 30.61 seconds (0.51 minutes)
```

In [ ]:
```

```