Semantic Structure using PropBank

In [1]:
```
import nltk
```

In [2]:
```
from nltk.corpus import treebank
```

In [3]:
```
pip install svgling
```

```
Requirement already satisfied: svgling in /home/user/anaconda3/lib/python3.10/site-packages (0.4.0)
Requirement already satisfied: svgwrite in /home/user/anaconda3/lib/python3.10/site-packages (from svgling) (1.4.3)
Note: you may need to restart the kernel to use updated packages.
```

In [4]:
```
treebank.parsed_sents()[0]
```

Out[4]:

In [5]:
```
from nltk.corpus import propbank
```

In [6]:
```
print(propbank.instances()[0])
```

```
wsj_0001.mrg 0 8 gold join.01 vf--a 0:2-ARG0 7:0-ARGM-MOD 8:0-rel 9:1-ARG1 11:1-ARGM-PRD 15:1-ARGM-TMP
```

In [7]:
```
pb_instances=propbank.instances()
```

In [8]:
```
print(pb_instances[42])
```

```
wsj_0003.mrg 15 19 gold say.01 vp--a 1:2*20:0-ARG1 19:0-rel 21:1-ARG0
```

In [9]:
```
print(pb_instances[103])
```

```
wsj_0004.mrg 8 16 gold rise.01 vp--a 0:2-ARG1 13:1-ARGM-DIS 16:0-rel 17:1-ARG4-to 20:1-ARG3-from
```

In [10]:
```
len(pb_instances)
```

Out[10]:
```
112917
```

In [11]:
```
len(propbank.verbs())
```

Out[11]:
```
3257
```

In [12]:
```
print(propbank.verbs()[:20])
```

```
['abandon', 'abate', 'abdicate', 'abet', 'abide', 'abolish', 'abort', 'abound', 'abridge', 'absolve', 'absorb', 'abst
ain', 'abuse', 'accede', 'accelerate', 'accept', 'access', 'acclaim', 'accommodate', 'accompany']
```

In [13]:
```
print(propbank.verbs()[-20:])
```

```
['wrap', 'wreak', 'wreck', 'wrench', 'wrest', 'wrestle', 'wriggle', 'wring', 'write', 'writhe', 'wrong', 'yank', 'yel
l', 'yelp', 'yield', 'zap', 'zero', 'zip', 'zone', 'zoom']
```

In [14]:
```
propbank.roleset('join.01')
```

Out[14]:
```
<Element 'roleset' at 0x78d76441be70>
```

In [15]:
```
for role in propbank.roleset('join.01').findall('roles/role'):
    print(role.attrib['n'], role.attrib['descr'])
```

```
0 agent, entity doing the tying
1 patient, thing(s) being tied
2 instrument, string
```

In [16]:
```
rise01=propbank.roleset('rise.01')
for role in rise01.findall('roles/role'):
    print(role.attrib['n'],role.attrib['descr'])
```

```
1 Logical subject, patient, thing rising
2 EXT, amount risen
3 start point
4 end point
M medium
```

In [17]:
```
abandon01=propbank.roleset('abandon.01')
for role in rise01.findall('roles/role'):
    print(role.attrib['n'],role.attrib['descr'])
```

```
1 Logical subject, patient, thing rising
2 EXT, amount risen
3 start point
4 end point
M medium
```

In [18]:
```
inst0=pb_instances[0]
print(inst0)
print(inst0.fileid,inst0.sentnum,inst0.wordnum,inst0.tagger)
inst0.tree
```

```
wsj_0001.mrg 0 8 gold join.01 vf--a 0:2-ARG0 7:0-ARGM-MOD 8:0-rel 9:1-ARG1 11:1-ARGM-PRD 15:1-ARGM-TMP
wsj_0001.mrg 0 8 gold
```

Out[18]:

In [19]:
```
print(dir(inst0))
```

```
['__class__', '__delattr__', '__dict__', '__dir__', '__doc__', '__eq__', '__format__', '__ge__', '__getattribute__',
'__gt__', '__hash__', '__init__', '__init_subclass__', '__le__', '__lt__', '__module__', '__ne__', '__new__', '__redu
ce__', '__reduce_ex__', '__repr__', '__setattr__', '__sizeof__', '__str__', '__subclasshook__', '__weakref__', '_get_
tree', 'arguments', 'baseform', 'fileid', 'inflection', 'parse', 'parse_corpus', 'predicate', 'predid', 'roleset', 's
ensenumber', 'sentnum', 'tagger', 'tree', 'wordnum']
```

In [20]:
```
inst0.arguments
```

Out[20]:
```
((PropbankTreePointer(0, 2), 'ARG0'),
 (PropbankTreePointer(7, 0), 'ARGM-MOD'),
 (PropbankTreePointer(9, 1), 'ARG1'),
 (PropbankTreePointer(11, 1), 'ARGM-PRD'),
 (PropbankTreePointer(15, 1), 'ARGM-TMP'))
```

In [21]:
```
for (argloc,argid) in inst0.arguments:
    print(argid)
    print(argloc.select(inst0.tree))
    print()
```

```
ARG0
(NP-SBJ
  (NP (NNP Pierre) (NNP Vinken))
  (, ,)
  (ADJP (NP (CD 61) (NNS years)) (JJ old))
  (, ,))

ARGM-MOD
(MD will)

ARG1
(NP (DT the) (NN board))

ARGM-PRD
(PP-CLR (IN as) (NP (DT a) (JJ nonexecutive) (NN director)))

ARGM-TMP
(NP-TMP (NNP Nov.) (CD 29))
```

In [ ]:

FrameSet using FramenNet

In [22]:
```
from pprint import pprint
```

In [23]:
```
from operator import itemgetter
```

In [24]:
```
from nltk.corpus import framenet as fn
```

In [25]:
```
from nltk.corpus.reader.framenet import PrettyList
```

In [26]:
```
x=fn.frames(r'(?i)crim')
```

In [27]:
```
x.sort(key=itemgetter('ID'))
x
```

Out[27]:
```
[<frame ID=200 name=Criminal_process>, <frame ID=500 name=Criminal_investigation>, ...]
```

In [28]:
```
PrettyList(sorted(x,key=itemgetter('ID')))
```

Out[28]:
```
[<frame ID=200 name=Criminal_process>, <frame ID=500 name=Criminal_investigation>, ...]
```

In [29]:
```
f=fn.frame(202)
f.ID
```

Out[29]:
```
202
```

In [30]:
```
f.name
```

Out[30]:
```
'Arrest'
```

In [31]:
```
len(f.lexUnit)
```

Out[31]:
```
11
```

In [32]:
```
pprint(sorted([x for x in f.FE]))
```

```
['Authorities',
 'Charges',
 'Co-participant',
 'Manner',
 'Means',
 'Offense',
 'Place',
 'Purpose',
 'Source_of_legal_authority',
 'Suspect',
 'Time',
 'Type']
```

In [33]:
```
pprint(f.frameRelations)
```

```
[<Parent=Intentionally_affect -- Inheritance -> Child=Arrest>, <Complex=Criminal_process -- Subframe -> Component=Arr
est>, ...]
```

In [34]:
```
x=fn.frames('crime')
```

In [35]:
```
x.sort(key=itemgetter('ID'))
x
```

Out[35]:
```
[<frame ID=700 name=Committing_crime>]
```

In [36]:
```
f=fn.frame(700)
f.name
```

Out[36]:
```
'Committing_crime'
```

In [ ]:
```

```