Basics

Loading LingPy

In [2]:
```
from lingpy import * # basic functionalities of LingPy
from lingpy.compare.phylogeny import * # borrowing detection class PhyBo
from lingpy.convert.plot import * # general plot module
from lingpy.evaluate.acd import * # evaluation module for cognate detection
from IPython.core.display import Image # only for ipython-rendering in this con
```

Loading Data

In [3]:
```
wl = Wordlist('BAI.qlc')
```

Checking Data

In [4]:
```
number_of_taxa = wl.width
number_of_concepts = wl.height
number_of_entries = len(wl)
print("Wordlist has {0} entries, distributed over {1} languages and {2} concept
```

```
Wordlist has 1028 entries, distributed over 9 languages and 110 concepts.
```

Retrieve data

In [5]:
```
ashes = wl.get_dict(concept='ashes', entry='ipa')
for a,b in sorted(ashes.items(), key=lambda x:x[0]):
    print("{0:10}".format(a),"\t",b[0])
```

```
Ega             xwa55lo21ɕu55
Enqi            xwi22ɕu24
Gongxing        xwɛ22ɕy55ʁwa12
Jinman          kʰwa55la21ɕu55
Jinxing         su55
Mazhelong       xwa31la44su55
Tuolo           ʂɤ55
Zhoucheng       su55
```

Manipulate data

In [6]:
```
msa = Multiple(sorted([v[0] for v in ashes.values()]))
msa.lib_align()
print(msa)
```

```
kʰ      w       a       55      l       a       21      ɕ       u
55      -       -       -       -
-       -       -       -       -       -       -       s       u
55      -       -       -       -
-       -       -       -       -       -       -       s       u
55      -       -       -       -
x       w       a       31      l       a       44      s       u
55      -       -       -       -
x       w       a       55      l       o       21      ɕ       u
55      -       -       -       -
x       w       i       22      -       -       -       ɕ       u
24      -       -       -       -
x       w       ɛ       22      -       -       -       ɕ       y
55      ʁ       w       a       12
-       -       -       -       -       -       -       ʂ       ɤ
55      -       -       -       -
```

Find Cognates

In [7]:
```
lex = LexStat('BAI.qlc')
print(', '.join([h for h in lex.header if h not in wl.header]))
```

```
sonars, weights, classes, duplicates, langid, prostrings, numbers
```

In [8]:
```
lex.get_scorer(ratio=(1,0), force=True)
```

In [9]:
```
lex.cluster(method='lexstat', threshold=0.6)
```

In [10]:
```
lex.export('txt', filename='lexstat', sections=dict(h1=("concept", "# Concept:
```

Look up the file lexstat.txt to see the results.

Align Cognate Sets

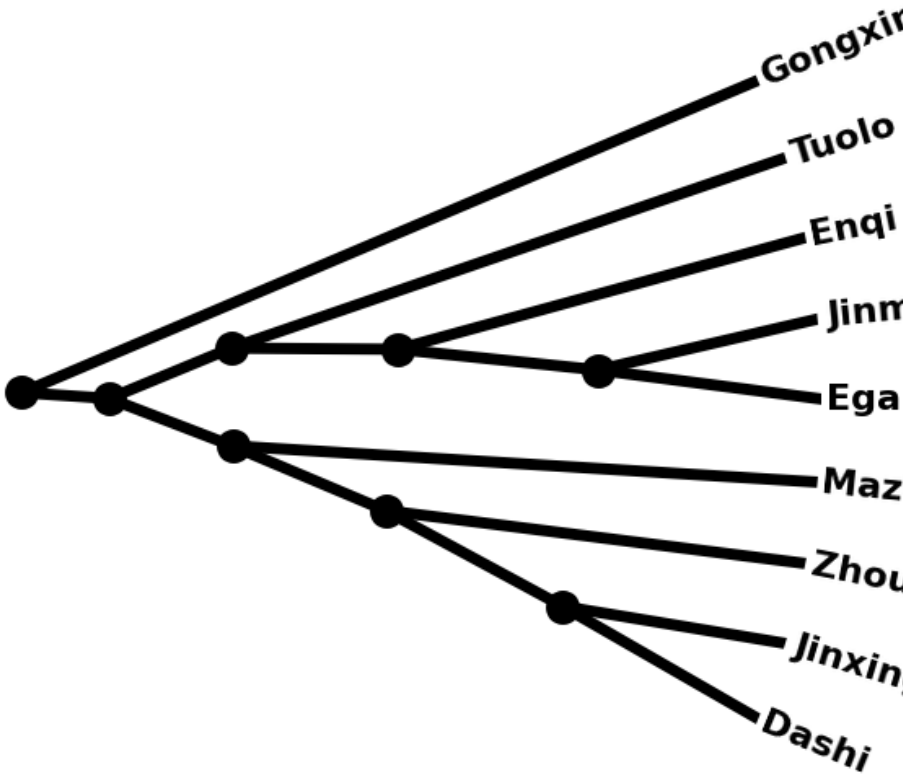
In [11]:
```
alm = Alignments(lex, ref="lexstatid")
alm.align(scoredict=lex.cscorer)
alm.output('html', filename='alignments')
```

Look up the file alignments.html to check the results.

Calculate Trees

In [12]:
```
alm.calculate('tree', ref='lexstatid', tree_calc="neighbor")
plot_tree(alm.tree, filename="bai", fileformat="png", bg="white", textcolor="bl
Image('bai.png')
```

Out[12]:



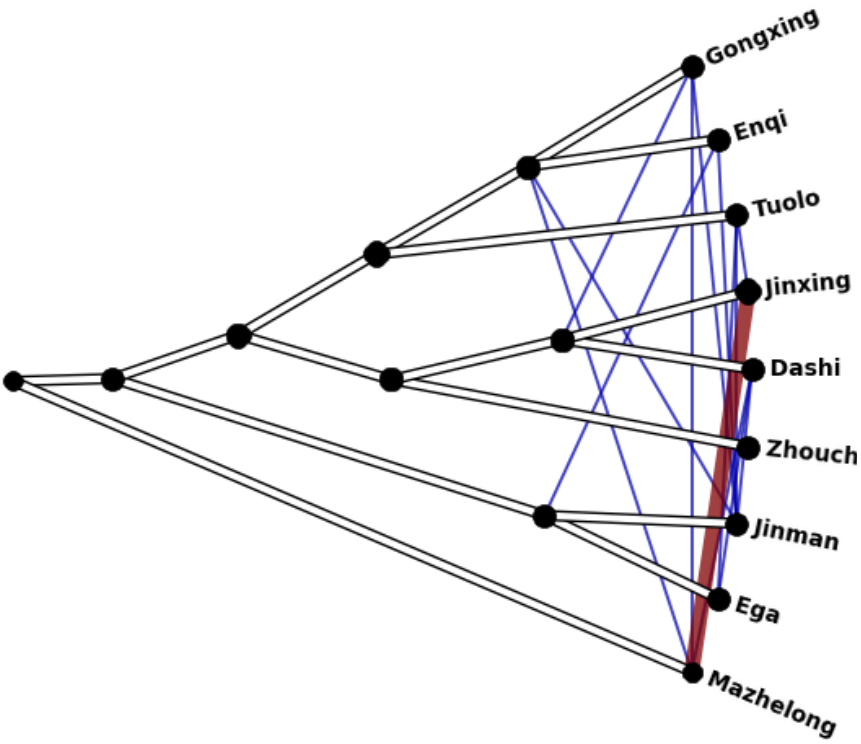
Find Borrowings

In [13]:
```
lex.output('qlc', filename="bai_lexstat")
phy = PhyBo('bai_lexstat.qlc', ref="lexstatid", degree=45, start=270, tree_calc
phy.analyze()
noo,pdc = phy.get_stats(phy.best_model)
print("Best model is {0} with {1:.2f} origins and {2:.2f} % of patchy cognates.
```

```
Best model is w-5-2 with 1.12 origins and 0.12 % of patchy cognates.
```

In [29]:
```
phy.get_MLN(phy.best_model)
phy.plot_MLN(phy.best_model, fileformat='png', filename='bai_mln', nodestyle="v
Image('bai_mln.png')
```

Out[29]:



In [14]:
```
D = {}
P = {}
for key in phy:
    pat = phy[key, "patchy"]
    if not pat.endswith("0"):
        tax = phy[key, "taxa"]
        wrd = phy[key, "ipa"]
        con = phy[key, "concept"]
        try:
            D[pat] += [(tax,con,wrd)]
        except:
            D[pat] = [(tax,con,wrd)]
        try:
            P[pat[:-2]] += [pat]
        except:
            P[pat[:-2]] = [pat]
        #print(tax,"\t", con, "\t", wrd, pat)

for pat in P:
    for patchy in sorted(set(P[pat])):
        for a,b,c in D[patchy]:
            print("{0:10}\t{1:10}\t{2:10}\t{3}".format(a,b,c,pat))
        print("")
    print("---\n")
```

```
Dashi           not             a42ɣẽ55         246:65

Jinman          not             a42             246:65
Enqi            not             a43             246:65
Ega             not             a42             246:65
Tuolo           not             a21             246:65

---

Dashi           say (V)         ʈaŋ21           297:75

Gongxing        say (V)         ʧu21            297:75
Tuolo           say (V)         zu42            297:75
Enqi            say (V)         ʈu21            297:75
Ega             say (V)         ʈo42            297:75

---

Tuolo           ashes           ʂɤ55            15:2

Zhoucheng       ashes           su55            15:2
Jinxing         ashes           su55            15:2

---

Ega             we              ŋwi55           422:102

Tuolo           we              we55            422:102

---

Mazhelong       knee            ko44ʈɛ44təɯ21   158:46

Jinman          knee            ko44ʈe44        158:46

---

Gongxing        cold            ga12            71:15

Dashi           cold            ka21            71:15
Jinxing         cold            kɑ21            71:15
```

Evaluating Cognate Detection Quality

In [15]:
```
a,b,c = bcubes(lex, 'cogid', 'lexstatid')
```

```
*****************************
* B-Cubed-Scores            *
* ------------------------- *
* B-Cubed-Precision: 0.9695 *
* B-Cubed-Recall:    0.8462 *
* B-Cubed-F-Scores:  0.9037 *
*****************************
```

In [16]:
```
lex.cluster(method='edit-dist', threshold=0.5)
a,b,c = bcubes(lex, 'cogid', 'editid')
```

```
*****************************
* B-Cubed-Scores            *
* ------------------------- *
* B-Cubed-Precision: 0.9785 *
* B-Cubed-Recall:    0.6701 *
* B-Cubed-F-Scores:  0.7955 *
*****************************
```

In [17]:
```
lex.cluster(method='turchin')
a,b,c = bcubes(lex, 'cogid', 'turchinid')
```

```
*****************************
* B-Cubed-Scores            *
* ------------------------- *
* B-Cubed-Precision: 0.9684 *
* B-Cubed-Recall:    0.6857 *
* B-Cubed-F-Scores:  0.8029 *
*****************************
```

In [18]:
```
lex.cluster(method='sca', threshold=0.4)
a,b,c = bcubes(lex, 'cogid', 'scaid')
```

```
[?] Datatype <scaid> has already been produced, do you want to override?
(y/n) y
*****************************
* B-Cubed-Scores            *
* ------------------------- *
* B-Cubed-Precision: 0.9577 *
* B-Cubed-Recall:    0.8850 *
* B-Cubed-F-Scores:  0.9199 *
*****************************
```

In [18]:
```

```