The Lancaster Corpus of Mandarin Chinese

Kyle Landin

Goal

- Make the data easier to read/more accessible
- Look specifically at the Parts of Speech tagset

Initial Attempt

xml would feed in each character into an individual column

In [7]:
```
#I want to move this list into a dataframe to make it more accessible
#Labels = ['Characters', 'Pinyin']
labels = ['0','1','2','3','4','5','6','7','8','9','10','11','12']
newsrep_df = pd.DataFrame.from_records(newsrep_chlist, columns=labels)
newsrep_df.head()
#Why are each individual character assigned their own column instead of being merged into one?
```

Out[7]:

	0	1	2	3	4	5	6	7	8	9	10	11	12
0	大	None	None	None	None	None	None	None	None	None	None	None	None
1	墻	None	None	None	None	None	None	None	None	None	None	None	None
2	内	外	None	None	None	None	None	None	None	None	None	None	None
3	北	京	市	None	None	None	None	None	None	None	None	None	None
4	监	狱	None	None	None	None	None	None	None	None	None	None	None

In [8]:
```
newsrep_df['Characters'] = newsrep_df[['0','1','2','3','4','5','6','7','8','9','10','11','12']].fillna('').sum(axis=1)
```

In [9]:
```
newsrep_df.head()
```

Out[9]:

	0	1	2	3	4	5	6	7	8	9	10	11	12	Characters
0	大	None	None	None	None	None	None	None	None	None	None	None	None	大
1	墻	None	None	None	None	None	None	None	None	None	None	None	None	墻
2	内	外	None	None	None	None	None	None	None	None	None	None	None	内外
3	北	京	市	None	None	None	None	None	None	None	None	None	None	北京市
4	监	狱	None	None	None	None	None	None	None	None	None	None	None	监狱

In [10]:
```
del newsrep_df['0']
del newsrep_df['1']
del newsrep_df['2']
del newsrep_df['3']
del newsrep_df['4']
del newsrep_df['5']
del newsrep_df['6']
del newsrep_df['7']
del newsrep_df['8']
del newsrep_df['9']
del newsrep_df['10']
del newsrep_df['11']
del newsrep_df['12']
#I'm sure there is a way to iterate this, but I was receiving errors trying to do it and this worked however crude it may be
```

In [64]:
```
newsrep_df.head()
newsrep_df.tail()
```

Out[64]:

	Characters
0	大
1	墻
2	内外
3	北京市
4	监狱

Initial Attempt

The pinyin column was especially bad

	0	1	2	3	4	5	6	7	8	9	...	31	32	33	34	35	36	37	38	39	Pinyin
0	d	a	4	None	None	None	None	None	None	None	...	None	None	None	None	None	None	None	None	None	da4
1	q	i	a	n	g	2	None	None	None	None	...	None	None	None	None	None	None	None	None	None	qiang2
2	n	e	i	4	w	a	i	4	None	None	...	None	None	None	None	None	None	None	None	None	nei4wai4
3	b	e	i	3	j	i	n	g	1	s	...	None	None	None	None	None	None	None	None	None	bei3jing1shi4
4	j	i	a	n	1	y	u	4	None	None	...	None	None	None	None	None	None	None	None	None	jian1yu4

```
In [16]: #Again, there has to be an easier way to do this
         del newsrep_pin_df['0']
         del newsrep_pin_df['1']
         del newsrep_pin_df['2']
         del newsrep_pin_df['3']
         del newsrep_pin_df['4']
         del newsrep_pin_df['5']
         del newsrep_pin_df['6']
         del newsrep_pin_df['7']
         del newsrep_pin_df['8']
         del newsrep_pin_df['9']
         del newsrep_pin_df['10']
         del newsrep_pin_df['11']
         del newsrep_pin_df['12']
         del newsrep_pin_df['13']
         del newsrep_pin_df['14']
         del newsrep_pin_df['15']
         del newsrep_pin_df['16']
         del newsrep_pin_df['17']
         del newsrep_pin_df['18']
         del newsrep_pin_df['19']
         del newsrep_pin_df['20']
         del newsrep_pin_df['21']
         del newsrep_pin_df['22']
         del newsrep_pin_df['23']
         del newsrep_pin_df['24']
         del newsrep_pin_df['25']
         del newsrep_pin_df['26']
         del newsrep_pin_df['27']
         del newsrep_pin_df['28']
         del newsrep_pin_df['29']
         del newsrep_pin_df['30']
         del newsrep_pin_df['31']
         del newsrep_pin_df['32']
         del newsrep_pin_df['33']
         del newsrep_pin_df['34']
         del newsrep_pin_df['35']
         del newsrep_pin_df['36']
         del newsrep_pin_df['37']
         del newsrep_pin_df['38']
         del newsrep_pin_df['39']
```

Second Attempt

Started out with making each dataset into a set of tokens

News Editorial Data

In [3]:
```
newsedit_char = (lcmc_char + 'LCMC_B.xml')
newsedit_infile = open(newsedit_char, 'r', encoding='utf8')
newsedit_contents = newsedit_infile.read()
newsedit_soup = bs.BeautifulSoup(newsedit_contents, 'xml')

nechar = [w_tag.text for w_tag in newsedit_soup.find_all('w')]
nepos = [w_tag.get('POS') for w_tag in newsedit_soup.find_all('w')]

newsedit_pin = (lcmc_pinyin + 'LCMC_B.xml')
newsedit_pin_infile = open(newsedit_pin, 'r', encoding='utf8')
newsedit_pin_contents = newsedit_pin_infile.read()
newsedit_pin_soup = bs.BeautifulSoup(newsedit_pin_contents, 'xml')

nepinyin = [w_tag.text for w_tag in newsedit_pin_soup.find_all('w')]

ne_tokens = list(zip(nechar, nepinyin, nepos))
ne_unique = set(ne_tokens)

ne_tokens[:5]
```

Out[3]:
```
[('缓解', 'huan3jie3', 'v'), ('南北', 'nan2bei3', 'f'), ('矛盾', 'mao2dun4', 'an'), ('的', 'de5', 'u'), ('出路', 'chu1lu4', 'n')]
```

Compiled Data

With all of the lists combined, there are 839,006 words total in the corpus (this includes numerals in some cases)

When all duplicate words are removed, this leaves 53,379 words

Compiled Data

Using the compiled data, we can look at things like the most common words, how many words end in 的, etc.

```
from collections import Counter

freq = Counter(Compiled_Data)
freq.most_common(20)
```

```
[(('的', 'de5', 'u'), 50832), (('是', 'shi4', 'v'), 11427), (('了', 'le5', 'u'), 10379), (('在', 'zai4', 'p'), 9899), (('一', 'yi1', 'm'), 8365), (('和', 'he2', 'c'), 7200), (('他', 'ta1', 'r'), 5847), (('不', 'bu4', 'd'), 5594), (('我', 'wo3', 'r'), 5575), (('有', 'you3', 'v'), 4957), (('这', 'zhe4', 'r'), 4142), (('人', 'ren2', 'n'), 4025), (('也', 'ye3', 'd'), 3895), (('说', 'shuo1', 'v'), 3668), (('上', 'shang4', 'f'), 3558), (('着', 'zhao2', 'u'), 3364), (('就', 'jiu4', 'd'), 3236), (('地', 'di4', 'u'), 3209), (('中', 'zhong1', 'f'), 3175), (('对', 'dui4', 'p'), 3155)]
```

```
de_final = [(w,p,pos) for (w,p,pos) in Compiled_Unique if w.endswith('的')]
len(de_final)
de_final
```

```
46
```

```
[('厚厚的', 'hou4hou4de5', 'z'), ('的的', 'de5de5', 'n'), ('鼓鼓的', 'gu3gu3de5', 'z'), ('淡淡的', 'dan4dan4de5', 'z'), ('别的', 'bie2de5', 'r'), ('长长的', 'chang2chang2de5', 'z'), ('的', 'de5', 'b'), ('短短的', 'duan3duan3de5', 'z'), ('小小的', 'xiao3xiao3de5', 'z'), ('真的', 'zhen1de5', 'a'), ('是的', 'shi4de5', 'y'), ('的', 'de5', 'vn'), ('死死的', 'si3si3de5', 'z'), ('的', 'de5', 'j'), ('端的', 'duan1de5', 'd'), ('目的', 'mu4di4', 'n'), ('细细的', 'xi4xi4de5', 'z'), ('众矢之的', 'zhong4shi3zhi1de5', 'i'), ('的', 'de5', 'v'), ('薄薄的', 'bo2bo2de5', 'z'), ('高高的', 'gao1gao1de5', 'z'), ('婊子养的', 'biao3zi5yang3de5', 'l'), ('的', 'de5', 'nr'), ('真的', 'zhen1de5', 'd'), ('的', 'de5', 'd'), ('微微的', 'wei2wei2de5', 'z'), ('静静的', 'jing4jing4de5', 'z'), ('有的', 'you3de5', 'r'), ('浓浓的', 'nong2nong2de5', 'z'), ('似的', 'si4de5', 'u'), ('的', 'de5', 'n'), ('当家的', 'dang1jia1de5', 'n'), ('的的', 'de5de5', 'v'), ('狗日的', 'gou3ri4de5', 'l'), ('小的', 'xiao3de5', 'r'), ('的', 'de5', 'u'), ('的', 'de5', 'f'), ('他娘的', 'ta1niang2de5', 'l'), ('好样儿的', 'hao3yang4er2de5', 'n'), ('别的', 'bie2de5', 'a'), ('他妈的', 'ta1ma1de5', 'l'), ('老不死的', 'lao3bu4si3de5', 'l'), ('妈x的', 'ma1Xde5', 'l'), ('你妈的', 'ni3ma1de5', 'l'), ('圆圆的', 'yuan2yuan2de5', 'z'), ('亲爱的', 'qin1ai4de5', 'n')]
```

Compiled Data

Count of unique words and their parts of speech