Abstract:
I took at the noun phrases from all the enrolled bills from the 106th, 107th, and 108th congresses. I then analyzed the introduction date of the bill to the House or Senate. Using this date (and some common sense) I grouped all bills into one of two groups - before 09/11/2001 and after. My hypothesis was that the language of bills introduced after 9/11 would become more "patriotic" or nationalistic. In an effort to sift through the massive number of phrases I used tf-idf ranking and only listed the top few noun phrases.

Data Set:
My data set was actually one I built a few years ago with perl scripts. It is the plain text of every official version of every bill from the Senate and the House from the 103rd congress to the 110th. I only linguistically analyzed enrolled (passed) bills from the 106th, 107th, and 108th congresses (1791 bills total).

Method:
My method was fairly simple. I used python to get a list of all the file names for enrolled bills in the target congresses. I then wrote a function to analyze the previous versions of that bill to find it's introduction date. Using that I split the corpus into a 'before' and an 'after' category. When I had the proper list of filenames, I fed them through the MontyLingua jist function to generate the list of noun phrases. I added the list of noun phrases to a dictionary that was keyed by the bill number. I then ran a tf-idf ranking on each subset in each congress, using the noun phrases as approximations of tokenized full-text. Since the entirety of the 106th congress was before (and the 108th after) I did not seperate the samples. I output the top tf-idf ranked noun phrases as my lists and visually analyzed the results.

Code:
```
#!/usr/bin/python

import glob
import math
import os
import pprint
import re
import sys

sys.path.insert(0, '/home/foo/montylingua-2.1/python')
import MontyLingua

def main():
    os.chdir('/home/foo/montylingua-2.1/python')
    ml = MontyLingua.MontyLingua()
    pp = pprint.PrettyPrinter(indent=2)
    sample_years = ('106th', '107th', '108th')
    output_file = open('/home/foo/nlprun.log', 'w')
    for cong in sample_years:
        bill_path = '/home/foo/%s/enr' % cong
        bill_files = glob.glob(bill_path + '/*.txt')

        (before, after) = GetDates(bill_files, cong)
        output_file.write('#get the noun phrases from each of these Bills.
        output_file.write('#list of all noun phrases in each file keyed by billno.'
```

before_np = GetNounPhrases(before, ml)
after_np = GetNounPhrases(after, ml)

(btf, atf, idf) = GetTfIdf(before_np, after_np)

before_top = []
after_top = []
allnp = []
for (bill, npd) in btf.iteritems():
    allnp += [(x['tfidf'], y) for (y, x) in npd.iteritems()]
allnp.sort()
before_top += [x for x in allnp[:100]]
allnp = []
for (bill, npd) in atf.iteritems():
    allnp += [(x['tfidf'], y) for (y, x) in npd.iteritems()]
allnp.sort()
after_top += [x for x in allnp[:100]]

output_file.write("for congress %s\n" % cong)
output_file.write("before words\n")
linecount = 1
for (score, phrase) in before_top:
    output_file.write("%s\%t\%.5f\%t\%s\n" % (linecount, score, phrase))
    linecount += 1
output_file.write("\n")
output_file.write("after words\n")
output_file.write("\n")

output_file.write("for congress %s\n" % cong)
output_file.write("before words\n")
output_file.write("\n")

def GetDates(bill_files, cong):
    '''Return the introduction and passage date of the bill.
    '''
    base_path = '/home/foo/%s/' % cong
    # all the bills passed introduced before 09/11/2001
    before = []
    # all the bills passed introduced after 09/11/2001
    after = []
    date_re = re.compile(r'''
([January|February|March|April|May|June|July|August|September|October|November|December]\s(\[\d\]{1,2}),\s(200[1-3]))''', re.X)
s_search_order = ('is', 'ats', 'rfh')
h_search_order = ('ih', 'rds', 'rfs', 'ath')
for file_path in bill_files:
    billno = file_path.split('/')[-1]
    billbase = billno[:-7]
    if billno[0] == 's':
       so = s_search_order
    else:
       so = h_search_order
    filex = False
    for suffix in so:
        if os.path.isfile(base_path+billbase+suffix+'.txt')
           and not filex:
            filex = True
            f_path = base_path+billbase+suffix+'.txt'
            match = date_re.search(''.join(open(f_path).readlines()))
            if match == None:
              print billno
              continue
            mon = match.group(1)
            yr = match.group(4)
            if int(match.group(4)) > 2001:
                after.append(file_path)
            elif ('Oct' in mon or 'Nov' in mon or 'Dec' in mon) and yr == 2001):
                after.append(file_path)
            elif ('Sep' in mon and match.group(3) > 11 and yr == 2001):
                after.append(file_path)
            else:
                before.append(file_path)
    return(before, after)

def GetNounPhrases(bill_list, ml):
    '''Get the monty lingua dict and return a {billno: [np, np, ...]}'''
    output = {}
    for bill in bill_list:
        billno = bill.split('/')[-1]
        b = billno[-7]
        output[b] = []
        for section in ml.jist(''.join(open(bill).readlines())):
            output[b] += [x.lower() for x in section['noun_phrases'] if len(x) > 2]
    return output

def GetTfIdf(before_np, after_np):
    '''Return the tf Ídf dictionaries.''
    #{np: idf-score, ...}
    idf_before = {}
    idf_after = {}
    idf = {}
    #{billno: (np: tf-score), ...}
tf_before = {}
tf_after = {}

phrases = {}
docs = float(len(before_np.keys()) + 
len(after_np.keys()))
for (bill, nplist) in before_np.iteritems():
    tf_before[bill] = {}
    tf_total = float(len(nplist))
    for np in nplist:
        if np not in phrases:
            phrases[np] = []
            phrases[np].append(bill)
        elif bill not in phrases[np]:
            phrases[np].append(bill)
        if np not in tf_before[bill]:
            tf_before[bill][np] = {}
            tf_before[bill][np]['count'] = 0
            tf_before[bill][np]['tf'] = 0
            tf_before[bill][np]['count'] += 1
    for (np, stats) in tf_before[bill].iteritems():
        if stats == tf_total:
            continue
        stats['tf'] = stats['count']/tf_total

for (bill, nplist) in after_np.iteritems():
    tf_after[bill] = {}
    tf_total = float(len(nplist))
    for np in nplist:
        if np not in phrases:
            phrases[np] = []
            phrases[np].append(bill)
        elif bill not in phrases[np]:
            phrases[np].append(bill)
        if np not in tf_after[bill]:
            tf_after[bill][np] = {}
            tf_after[bill][np]['count'] = 0
            tf_after[bill][np]['tf'] = 0
            tf_after[bill][np]['count'] += 1
    for (np, stats) in tf_after[bill].iteritems():
        if stats == tf_total:
            continue
        stats['tf'] = stats['count']/tf_total
for np in phrases:
    idf[np] = math.log(docs/len(phrases[np]))

for data in (tf_before, tf_after):
    for bill in data:
        for (np, stats) in data[bill].iteritems():
            stats['tfidf'] = stats['tf'] * idf[np]

return (tf_before, tf_after, idf)
if __name__ == '__main__':
    main()

Data: (truncated somewhat for sanity)

for congress 106th
before words
# Score Phrase
1 0.01333 , section 1
2 0.01333 5284 one hundred sixth congress
3 0.01333 city
4 0.01333 congress
5 0.01333 designation
6 0.01333 docid
7 0.01333 effect
8 0.01333 effective date
9 0.01333 h5284enr
10 0.01333 house
11 0.01333 january , two thousand
12 0.01333 january 3 , 2001
13 0.01333 law , map , regulation , document , paper , or other record
14 0.01333 monday
15 0.01333 pre
16 0.01333 references
17 0.01333 second session begun
18 0.01333 section 1
19 0.01333 senate
20 0.01333 senate and house
21 0.01333 speaker
22 0.01333 twenty-fourth day
23 0.01333 txt
24 0.01333 united states and president
25 0.01333 vice president
26 0.01333 washington
27 0.02666 101 east main street
28 0.02666 act
29 0.02666 america
30 0.02666 norfolk , virginia
31 0.02666 reference
32 0.02666 representatives
33 0.02666 sec
34 0.03999 owen b.

for congress 107th
before words
# Score Phrase
1 0.00492 , section 1
2 0.00492 --subject
3 0.00492 --the boundary
4 0.00492 --the map
5 0.00492 --the secretary
6 0.00492 0 acres
7 0.00492 177
technical boundary adjustment
technical corrections
term
twenty-third day
txt
united states and president
utah wilderness act
vice president
wash canyon
wash canyon"
washington
wednesday
willow north a"
willow north b"
willow south"

531
8 acres
agriculture
america
boundary
file
lands
representatives
united states
act
sec
willow
4 acres
mount nebo wilderness
land
map

after words
# Score Phrase
1 0.00912  , section 1
2 0.00912  --section 1101
3 0.00912  --the secretary
4 0.00912  16 u.s.c.
5 0.00912  1978
6 0.00912  2002"
7 0.00912  3858 one hundred seventh congress
8 0.00912  beauty mountain , fayette county , west virginia
9 0.00912  boundary
10 0.00912  boundary modification
11 0.00912  city
12 0.00912  completion
13 0.00912  congress
14 0.00912  docid
15 0.00912  fee simple land exchange
16 0.00912  h3858enr
17 0.00912  house
18 0.00912  interior
19 0.00912  january , two thousand and two
20 0.00912  land exchange
21 0.00912  lands
22 0.00912  march 1996"
may 2001

national parks and recreation act

new river gorge boundary act

new river gorge national river, west virginia

new river gorge national river boundary modifications

pre

sec

second session begun

senate

senate and house

short title

speaker

such boundary

such land

that

treatment

twenty-third day

txt

united states and president

vice president

vicinity

washington

wednesday

act

america

net tract number

part

representatives

tract

boundaries

exchange

new river gorge national river

land

united states

for congress 108th

after words

# Score Phrase
1 0.01507  , section 1
2 0.01507 3011 one hundred eighth congress
3 0.01507 act
4 0.01507 city
5 0.01507 congress
6 0.01507 designation
7 0.01507 docid
8 0.01507 first session begun
9 0.01507 h3011enr
10 0.01507 house
11 0.01507 january , two thousand and three
12 0.01507 law , map , regulation , document , paper , or other record
13 0.01507 pre
14 0.01507 references
15 0.01507 sec
Conclusions:

Noun phrases don't make a good proxy for tokenized text in tf-idf weighting. While there is some shift to mentioning nationalistic nouns like 'america' and 'president' more frequently, it's not too pronounced. If I were to spend more time on this I would need to clean up the tokenizing pipeline. Probably adding a generous list of stop words.