Topic models from Mallet over text of turns

Running Mallet on the text of turns and considering them in a similar task setting as the unigram baseline for predicting app-user reactions.

Text and reactions

First, some text of turns to play with.

In [77]:
```
import pandas as pd
import reactions
import nltk
import random
import matplotlib.pyplot as plt
import os
from pandas.tools.plotting import scatter_matrix
```

The raw transcript and reaction info not yet grouped into turns.

In [9]:
```
%time r = reactions.link_reactions_to_transcript('data/reactions_oct3_4project.csv','corpora
```

```
CPU times: user 8.59 s, sys: 0.49 s, total: 9.07 s
Wall time: 9.08 s
```

In [50]:
```
print ' '.join(r.columns)
r2 = r.copy()
del r2["Sync'd start"]
del r2["Sync'd end"]
del r2["Time"]
del r2["Speaker"]
```

```
Frame QuestionTopic Reaction_what Reaction_who Speaker Sync'd end Sync'd start Time Tone
Topic Transcript UserID start statement turn Speaker_name
```

In [48]:
```
%time p = reactions.split_reactions_file('data/reactions_oct3_4project.csv')['quest_politica
' '.join(p.columns)
```

```
CPU times: user 4.79 s, sys: 0.35 s, total: 5.14 s
Wall time: 5.12 s
```

Out[48]:
```
'UserID party_1 political_views_2 candidate_choice_3 confidence_in_choice_4
likely_to_vote_5 immigration_priority_6 health_care_priority_7 foreign_policy_priority_8
abortion_priority_9 economy_priority_10 immigration_party_11 health_care_party_12
foreign_policy_party_13 abortion_party_14 economy_party_15 interested_23 news_sources_24
economy_candidate_27 foreign_policy_candidate_28 candidate_preferred_29'
```

In [13]:
```
p2 = p[['UserID','party_1','political_views_2','candidate_choice_3','confidence_in_choice_4'
p2['party'] = p2.party_1.apply(lambda a: {'closest to democratic party':'democrat',
                                          'lean democrat':'democrat',
                                          'lean republican':'republican',
                                          'closest to republican party':'republican'}.get(a,
```

In [47]:
```
%time r3 = r2.merge(p2[['UserID','party']])
```

```
' '.join(r3.columns)
```

```
CPU times: user 0.63 s, sys: 0.04 s, total: 0.67 s
Wall time: 0.67 s
```

Out[47]:
```
'Frame QuestionTopic Reaction_what Reaction_who Tone Topic Transcript UserID start
statement turn Speaker_name party'
```

Group by turns

In [15]:
```
st = r3.groupby(['statement']).first()[['Speaker_name','Transcript','turn']]
t = pd.DataFrame({'speaker':st.groupby('turn').first().Speaker_name,
                  'reactions':r3.groupby('turn').count().Speaker_name,
                  'statements':st.groupby('turn').count().turn,
                  'text':st.groupby('turn').apply(lambda x: ''.join(x.Transcript)),
                  'agree':r3[r3.Reaction_what=='Agree'].groupby('turn').count().turn,
                  'agree_dem':r3[(r3.party=='democrat') & (r3.Reaction_what=='Agree')].group
                  'agree_rep':r3[(r3.party=='republican') & (r3.Reaction_what=='Agree')].gro
                  'disagree':r3[r3.Reaction_what=='Disagree'].groupby('turn').count().turn,
                  'disagree_dem':r3[(r3.party=='democrat') & (r3.Reaction_what=='Disagree')]
                  'disagree_rep':r3[(r3.party=='republican') & (r3.Reaction_what=='Disagree'
                  })
t['words'] = t.text.apply(lambda txt: [t.lower() for t in nltk.tokenize.word_tokenize(txt) i
t['word_count'] = t.words.apply(lambda words: len(words))
t['r_per_st'] = 1.0 * t.reactions / t.statements
t['r_per_w'] = 1.0 * t.reactions / t.word_count
t['a_to_d_dems'] = t.agree_dem / t.disagree_dem
t['a_to_d_reps'] = t.agree_rep / t.disagree_rep
```

Unigram features.

In [37]:
```
ranked_unigrams = nltk.FreqDist([w for word_list in t.words for w in word_list]).keys()
MAX_FEATURES = 700 # avoid overfitting
t['unigrams'] = t.words.apply(lambda words: {w:True for w in words if w in ranked_unigrams[:
t['unigram_count'] = t.unigrams.apply(lambda unigrams: len(unigrams))
```

Remove short turns.

In [16]:
```
MIN_WORDS = 30
t2 = t[t.word_count >= MIN_WORDS]
print len(t),'->',len(t2)
```

```
190 -> 71
```

Label turns.

In [17]:
```
t2['label'] = t2.a_to_d_dems >= t2.a_to_d_dems.quantile(.5)
```

In [46]:
```
' '.join(t2.columns)
```

Out[46]:
```
'agree agree_dem agree_rep disagree disagree_dem disagree_rep reactions speaker statements
text words word_count r_per_st r_per_w a_to_d_dems a_to_d_reps unigrams unigram_count
label'
```

LDA features

Write text to input file for Mallet.

A folder for holding Mallet files for this topics vs unigram comparison.

In [213]:
```
!rm -r tmp.tVSu
!mkdir tmp.tVSu
```

In [214]:
```
f = open('tmp.tVSu/in.txt','w')
for i,r in t2.iterrows():
    f.write("%d NA %s\n" % (i, ' '.join(r['words'])))
f.close()
```

In [245]:
```
!head -n 1 tmp.tVSu/in.txt
```

```
1 NA good evening from the magness arena at the university of denver in denver jim lehrer
of the pbs newshour and i welcome you to the first of the presidential debates between
president barack obama the democratic nominee and former massachusetts governor mitt romney
the republican debate and the next three two presidential one presidential are sponsored by
the commission on presidential minutes will be about domestic issues and will follow a
format designed by the will be six roughly segments with answers for the first question
then open discussion for the remainder of each of people offered suggestions on segment
subjects or questions via the internet and other means but i made the final selectionsand
for the record they were not submitted for approval to the commission or the segments as i
announced in advance will be three on the economy and one each on health care the role of
government and governing with an emphasis throughout on differences specifics and
candidates will also have closing audience here in the hall has promised to remain cheers
applause boos hisses among other noisy distracting things so we may all concentrate on what
the candidates have to is a noise exception right now though as we welcome president obama
and governor gentlemen welcome to you start the economy segment let begin with are the
major differences between the two of you about how you would go about creating new jobs you
have two minutes each of you have two minutes to coin toss has determined president you go
first
```

Use Mallet to prep the input file before training topics.

In [227]:
```
!mallet/bin/mallet import-file --input tmp.tVSu/in.txt --output tmp.tVSu/in.mallet --keep-se
!ls tmp.tVSu/
```

```
in.mallet   in.txt     keys.txt    log.txt     state.gz    topics.txt
```

Train topics.

In [228]:
```
!mallet/bin/mallet train-topics --input tmp.tVSu/in.mallet --num-topics 30 --output-state tm
```

Wait until output files appear.

In [222]:
```
!ls tmp.tVSu/
```

```
in.mallet   in.txt     keys.txt    log.txt     state.gz    topics.txt
```

In [247]:
```
!head -n 2 tmp.tVSu/topics.txt
```

```
#doc name topic proportion ...
0       1       13      0.45926618183203816     8       0.35710755777295733     24
0.0788980922235202      23      0.04730063985040274     28      0.029975056448629345    4
0.009686990895312646    21      0.004341785068119908    3       0.0014216965243021948   14
0.0013318089480645195   15      0.0012738606858953686   20      0.0010705219464362 4    12
9.539719251060424E-4    11      7.910957781564223E-4    25      5.310775209234101E-4    2
```

```
5.305902722393401E-4	6	5.30106004709156E-4	26	5.286668084779E-4	1
4.244008874088659E-4	17	4.0297150830554166E-4	19	4.0261376798926167E-4	22
3.700997635907186E-4	0	3.6937669411317057E-4	5	3.6620073986661167E-4	29
3.621127098850232E-4	27	3.5949360578337887E-4	7	3.229038652824974E-4	10
3.217396348623458E-4	18	2.6964341416511753E-4	16	2.6619109832790326E-4	9
2.2255180512863995E-4
```

Read Mallet output.

In [271]:
```
turns = []
feats = []
header = True
for line in open('tmp.tVSu/topics.txt','r'):
    if header:
        header = False
        continue
    fs = line.split('\t')
    turns.append(int(fs[1]))
    feats.append({int(f):float(v) for f,v in zip(fs[2::2], fs[3::2])})
#mo = pd.DataFrame({'turn':turns, 'topics':feats})
mo = pd.DataFrame({'topics':feats}, index=pd.Int64Index(turns,name='turn'))
#mo = mo.set_index('turn')
mo.head(2)
```

Out[271]:

	topics
turn	
1	{0: 0.00036937669411317057, 1: 0.0004244008874...
2	{0: 0.0003875201854763252, 1: 0.01026908909979...

Merge topic features with other data

In [282]:
```
t3 = t2.join(mo)
t3[['text','label','unigrams','topics']].head(2)
```

Out[282]:

	text	label	unigrams	topics
turn				
1	Good evening from the Magness Arena at the Uni...	True	{'all': True, 'domestic': True, 'questions': T...	{0: 0.00036937669411317057, 1: 0.0004244008874...
2	Well, thank you very much, Jim, for this oppor...	True	{'sector': True, 'all': True, 'code': True, 'j...	{0: 0.0003875201854763252, 1: 0.01026908909979...

Experiment with classification

In [313]:
```
ex = t3
```

In [314]:
```
train_rows = random.sample(ex.index, len(ex)*9/10)
trn = ex.ix[train_rows]
tst = ex.drop(train_rows)
print len(trn)
```

```
print len(tst)
```

```
63
8
```

In [315]:
```
%time cl = nltk.NaiveBayesClassifier.train(zip(trn.topics, trn.label))
```

```
CPU times: user 0.02 s, sys: 0.00 s, total: 0.02 s
Wall time: 0.02 s
```

In [316]:
```
nltk.classify.accuracy(cl, zip(tst.topics, tst.label))
```

Out[316]:
```
0.625
```

In [317]:
```
cl.show_most_informative_features(10)
```

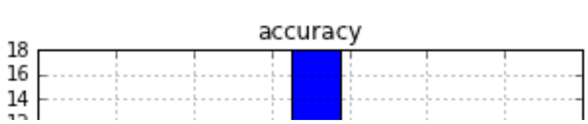
```
Most Informative Features
                             22 = 0.003089471295239396  False : True   =      1.7 : 1.0
                              9 = 0.001857789388941636  False : True   =      1.7 : 1.0
                              4 = 0.0026970996127855377  False : True   =      1.7 : 1.0
                             13 = 0.0036340230223573552  False : True   =      1.7 : 1.0
                              7 = 0.0026954954340779884  False : True   =      1.7 : 1.0
                              5 = 0.003056923525529371  False : True   =      1.7 : 1.0
                             25 = 0.004433260752509495  False : True   =      1.7 : 1.0
                             12 = 0.007963444371012015  False : True   =      1.7 : 1.0
                             15 = 0.010633770702863553  False : True   =      1.7 : 1.0
                              6 = 0.004425150854173767  False : True   =      1.7 : 1.0
```

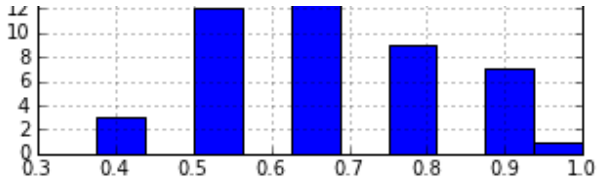
Multiple train/test

In [292]:
```
ex2 = t3
```

In [327]:
```
accs = []
for i in range(50):
    train_rows2 = random.sample(ex2.index, len(ex2)*9/10)
    trn2 = ex2.ix[train_rows2]
    tst2 = ex2.drop(train_rows2)

    # ~65%
    cl2 = nltk.NaiveBayesClassifier.train(zip(trn2.topics, trn2.label))

    # error..
    #cl2 = nltk.classify.MaxentClassifier.train(zip(trn2.topics, trn2.label))

    # ~50%
    #cl2 = nltk.classify.DecisionTreeClassifier.train(zip(trn2.topics, trn2.label))

    # error..
    #cl2 = nltk.classify.ConditionalExponentialClassifier.train(zip(trn2.topics, trn2.label)

    accs.append(nltk.classify.accuracy(cl2, zip(tst2.topics, tst2.label)))
print mean(accs)
a2 = pd.DataFrame({'accuracy':accs})
figsize(5,2)
a2.accuracy.hist()
title('accuracy')
show()
```

```
0.645
```





In [ ]: