1 Appendix A: Cluster Identification and Transition Analysis in Python

This code takes messages that are on twitter, and extracts their hashtags. It then constructs a set of weighted and unweighted network structures based upon co-citation of hashtags within a tweet. The network diagrams are interpreted to have a set of clusters within them which represent 'conversations' that are happening in the pool of twitter messages. We track similarity between clusters from day to day to investigate how conversations develop.

1.1 Utilities

These scripts depend upon a number of external utilities as listed below:

```
import json
import gzip
from collections import Counter
from itertools import combinations
import glob
import dateutil.parser
import pandas as pd
import os
import numpy as np
import datetime
import pickle
import subprocess
```

1.2 Data Files

We have twitter messages saved as compressed files, where each line in the file is the JSON object that the twitter sample stream returns to us. The files are created by splitting the streaming dataset according to a fixed number of lines - not necessarily by a fixed time or date range. A description of the collection process can be found in Appendix D.

All the files have the format `posts_sample_YYYYMMDD_HHMMSS_aa.txt` where the date listed is the date at which the stream was initialized. Multiple days worth of stream may be grouped under the same second, as long as the stream remains unbroken. If we have to restart the stream, then a new datetime will be added to the files.

1.3 Step 1: Count hashtag pairs

The most data-intensive part of the analysis is this first piece, which parses all of the input files, and counts the various combinations of hashtags on each day.

In this demonstration we perform this counting in memory, which is sufficient for date ranges on the order of weeks, but becomes unwieldy beyond this timescale.

```
#construct a counter object for each date
tallydict = dict([(date, Counter()) for date in dates])

#iterate through each of the input files in the date range
for i, zfile in enumerate(files):
    if i%10 == 0: #save every 10 files
        print i,
        with open(config['python_working_dir']+"tallydict.pickle", "wb"
            pickle.dump(tallydict, picklefile)
        with open(config['python_working_dir']+"progress.txt", 'a') as 
            progressfile.write(str(i)+': '+zfile+'\n')

    try:
        with gzip.open(zfile) as gzf:
            #look at each line in the file
            for line in gzf:
                try:
                    #parse the json object
                    parsed_json = json.loads(line)
                    # we only want to look at tweets that are in
                    # english, so check that this is the case.
                    if parsed_json.has_key('lang'):
                        if parsed_json['lang'] =='en':
                            #look only at messages with more than two h
                            #as these are the only ones that make conne
                            if len(parsed_json['entities']['hashtags'])
                        #extract the hashtags to a list
                        taglist = [entry['text'].lower() for en
                                 parsed_json['entities']['has
                        # identify the date in the message
                        # this is important because sometimes m
                        # come out of order.
                        date = dateutil.parser.parse(parsed_jso
                        date = date.strftime("%Y%m%d")
                        #look at all the combinations of hashta
                        for pair in combinations(taglist, 2):
                            #count up the number of alpha sorte
                            tallydict[date]['␣'.join(sorted(pai
                except: #error reading the line
                    print 'd',
        except: #error reading the file
            print 'error␣in', zfile
```